Techniques avancées d'optimisation pour la résolution du problème de stockage de conteneurs dans un port

Imen Ayachi Hajjem

N° d'ordre : 176

ÉCOLE CENTRALE DE LILLE
UNIVERSITÉ DE TUNIS EL MANAR
ÉCOLE NATIONALE D'INGÉNIEURS DE TUNIS

THÈSE

Présentée en vue d'obtenir le grade de

DOCTEUR

en

Spécialité : Automatique, Génie Informatique, Traitement du Signal et Image

Par

Imen AYACHI HAJJEM

Ingénieur – INSAT

Doctorat délivré conjointement par l'École Centrale de Lille et l'École Nationale d'Ingénieurs de Tunis

Titre de la thèse :

TECHNIQUES AVANCEES D'OPTIMISATION POUR LA RESOLUTION DU PROBLEME DE STOCKAGE DE CONTENEURS DANS UN PORT

Soutenue le 2 Mars 2012 devant le jury composé de :

M. Imed	KACEM	Professeur UDL, France	Président
M. Khaled	GHEDIRA	Professeur ISG, Tunis	Rapporteur
M. Abdellah	EL MOUDNI	Professeur UTBM, France	Rapporteur
Mme Hanen	BOUCHRIHA	Professeur ENIT, Tunis	Examinatrice
M. Pierre	BORNE	Professeur EC- Lille, France	Directeur de thèse
M. Mekki	KSOURI	Professeur ENIT, Tunis	Directeur de thèse

Thèse préparée dans le laboratoire LAGIS de l'École Centrale de Lille et au LACS de l'École Nationale d'Ingénieurs de Tunis, sous la direction des Professeurs P.Borne et M. Ksouri

Ecole Doctorale SPI 072 (Lille I, Lille III, Artois, ULCO, UVHC, EC Lille)

PRES Université Lille Nord-de-France

Dédicaces

Je dédie ce travail

A la mémoire de mon père.

A ma mère,

A mon cher époux,

A mon bonheur et ma joie de vie Ghazal

A mes chers sœur et frères

A ma belle mère

Qui m'ont toujours accordé leurs amours

Et leurs sacrifices.

A tous mes amis

Imen

Remerciements

Cette thèse est l'aboutissement de travaux réalisés dans le cadre d'une coopération et une cotutelle entre le laboratoire d'Automatique, Génie Informatique et Signal (LAGIS) de l'Ecole Centrale de Lille et le laboratoire d'Analyse et Commande des Systèmes (LACS) de l'Ecole Nationale d'Ingénieurs de Tunis.

Elle a été effectuée sous la co-direction du Professeur Pierre BORNE, du Professeur Mekki KSOURI et Docteur Ryan KAMMARTI.

Mes premiers remerciements vont particulièrement à M. Pierre BORNE, Professeur à l'EC-Lille, pour sa confiance, sa disponibilité et son soutien moral et scientifique tout au long de ce travail. Je le remercie pour ses qualités humaines et scientifiques.

Mes remerciements vont aussi à mon professeur M. Mekki KSOURI, Professeur à ENIT pour son aide, ses conseils et ses encouragements. Je tiens aussi à le remercier pour ses qualités humaines et son soutien permanent.

Je tiens à exprimer toute ma reconnaissance *à* M. Ryan KAMMARTI*, maître assistant à l'ISI, pour sa patience* sa perspicacité et ses précieux conseils qui ont contribué à la réalisation de ce travail. Merci de m'avoir soutenue pendant les moments difficiles. Un ami plus qu'un encadreur de thèse.

Je tiens à remercier M. Imed KACEM, Professeur à l'université de Lorraine, pour l'honneur qu'il m'a fait en acceptant d'être président de mon jury de thèse.

Un témoignage de ma profonde reconnaissance s'adresse à M. Khaled GHEDIRA, Professeur à l'institut supérieur de gestion (ISG) et à M. Abdellah EL MOUDNI, Professeur à l'Université de Technologie de Belfort Montbéliard, pour avoir accepté de rapporter ma thèse et pour leurs remarques et critiques constructives qui ont permis d'améliorer la qualité de ce manuscrit.

Mes vifs remerciements s'adressent pareillement à Madame Hanen BOUCHRIHA, Professeur à l'ENIT, qui a marqué son intérêt pour mon travail en acceptant d'être examinateur de cette thèse et en participant à ce jury.

Je présente ma profonde gratitude à toutes les personnes qui ont contribué directement ou indirectement à la réalisation de cette thèse, en particulier tout le personnel du LAGIS et les membres du LACS.

Merci à mes amis : Ammouna (Imen HARBAOUI), Hanoun (Hanène NAJJAR), Olfa (Olfa BEN AHMED), Mariouma(Meriem BRIKI), Hemden (Hajer BEN MAHMOUD), Noussa (Ines HAMMAMI) pour leurs encouragements et leurs disponibilités.

Merci à Lotfi, pour son soutien moral et pour sa présence inestimable à mes cotés.

Tables des matières

Liste des figures

Liste des figures

Liste des tableaux

Les publications des travaux de thèse

- Ayachi, I., Kammarti, R., Ksouri, M. and Borne, P., 2011, Application of Harmony search to Container Storage Location, IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2011), Anchorage, Alaska.

- Ayachi, I., Kammarti, R., Ksouri, M. and Borne, P., 2010, A Genetic algorithm to solve the container storage space allocation problem, IEEE Trans. International conference on Computational Intelligence and Vehicular System, pp 1-4, Seoul, South Korea.

- Ayachi, I., Kammarti, R., Ksouri, M. and Borne, P., 2010, Harmony search algorithm for the container storage problem, 8th International Conference of Modeling and Simulation - MOSIM'10, Tunisia.

- Kammarti, R., Ayachi, I., Ksouri, M. and Borne, P., 2009, Evolutionary Approach for the Containers Bin-Packing Problem, Studies in Informatics and Control, Vol. 18, No. 4/2009, pages 315-324.

Introduction Générale

L'homme n'existe que pour être dépassée.
Qu'avez-vous fait pour le dépasser ?
[Friedrich Nietzsche]

Introduction générale

Le transport maritime est le mode de transport de marchandises le plus utilisé et le plus important pour différentes raisons :

- Il est un moyen de transport peu coûteux (il coûte trente fois moins cher que le transport terrestre).
- Il offre des garanties de sûreté maximale pour les chargements de marchandises.
- Il permet l'acheminement des marchandises en grande masse.

Le transport maritime est un moyen de transport qui convient également pour des petits lots et de courtes distances. De ce fait, il peut être considéré comme un instrument privilégié des échanges internationaux. C'est pour cela qu'il a connu plusieurs révolutions pour s'adapter, au fil du temps, à l'évolution des échanges. L'une des révolutions les plus marquantes du transport maritime est la conteneurisation.

La conteneurisation est l'utilisation des conteneurs dans le transport des marchandises et ceci principalement dans le transport maritime. Ce processus a débuté dans les années 1960 et il est globalisé dans les années 80. Il vise à réduire les coûts de transport maritime en minimisant les coûts de manutention.

En effet, au lieu de charger / décharger chaque pièce du point de transport vers ou à partir d'un navire, la conteneurisation consiste à stocker des marchandises dans des conteneurs de différentes tailles normalisées. Le but de ce processus est de faciliter le stockage et améliorer l'efficacité et la rapidité des transports en réduisant les exigences d'emballage et de manutention dans tous les points de transfert entre le port et les voies ferroviaires et routières.

La conteneurisation a participé amplement à l'augmentation du trafic maritime mondial et les chiffres le démontrent bien. À la fin de 2005, la flotte mondiale de conteneurs marque une augmentation remarquable. Elle s'est chiffrée à 21,6 millions TEUs (Twenty feet Equivalent Units) ou EVP (équivalent vingt pied) et ce chiffre ne cesse d'augmenter (UNCTAD (2006)).

De plus, la capacité des navires est en croissance continue ; la première génération avait une capacité de 1700 TEUs et elle atteint aujourd'hui une capacité de 11000 TEUs. Les experts prédisent que la taille des navires continuera à augmenter avec un taux de 25% annuel jusqu'en 2013 (UNCTAD 2009). Donc, on peut estimer que les pays démunis des services de transport unitaires adéquats seront désavantagés dans leur commerce international (Castro (1999)).

Cependant, le déploiement de ces grands navires exige un investissement énorme d'infrastructures portuaires en fournissant un espace de stationnement plus grand et des grues plus puissantes.

Pour un fonctionnement efficace, les ports ont aussi besoin d'une zone de stockage spacieuse et d'une meilleure infrastructure routière et ferroviaire. Ceci soumet les opérateurs portuaires à une grande pression vu qu'ils doivent augmenter le débit du port et assurer la fiabilité et la qualité de leurs services tout en maintenant des coûts très bas.

Pour faire face à ce problème, deux types de solutions sont envisageables. La première consiste à améliorer la performance des ressources matérielles et spatiales du port par l'achat d'équipements de manutention puissants et par l'augmentation de la taille des ports ou même des espaces réservés au stationnement des navires et au stockage des conteneurs. Cette solution est facile à implémenter mais très coûteuse. La deuxième solution est stratégique, elle consiste à améliorer la productivité et la qualité des opérations portuaires en gardant la même structure du port maritime. Cette alternative consiste à étudier les principaux facteurs qui influencent la performance d'un terminal à conteneurs, en occurrence les stratégies d'exploitation et d'organisation du terminal, la planification des navires, la gestion des équipements de manutention, les exigences clientèle,…

Lors de la dernière décennie, cette voie nommée aussi logistique des terminaux à conteneurs a attiré beaucoup de chercheurs. En effet, de nombreuses simulations et de nombreux modèles analytiques ont été proposés pour résoudre différentes variantes de problèmes de gestion dans un terminal à conteneur. Parmi ceux-ci, on peut citer les problèmes de gestion des postes d'amarrage, de planification des grues de quai, de stockage et de transfert des conteneurs, etc.

Au cours de cette thèse, nous nous intéressons à l'optimisation du processus de stockage des conteneurs dans les différents blocks disponibles dans un terminal. Les différentes variantes de ce problème, traitées dans ce rapport, sont décrites dans le paragraphe suivant.

La gestion des conteneurs dans un port est une tâche très complexe. Elle consiste à trouver un arrangement « optimal » des conteneurs dans les emplacements disponibles des différents blocks d'un terminal. L'objectif majeur de ce problème est de minimiser le nombre de conteneurs à manipuler ou à déplacer lors du processus de déchargement.

Ce problème est généralement étudié en considérant un seul type de conteneur. Cependant, plusieurs types de conteneurs sont utilisés dans les ports maritimes à savoir les conteneurs généraux, réfrigérés, à toit ouvert, plate forme, ventilé, citerne, etc.

La plupart des chercheurs ont négligé la considération de plusieurs types de conteneurs or l'apparition de nouvelles contraintes de stockage relatives au stockage de chaque type rend le problème beaucoup plus compliqué.

En outre, le problème de stockage de conteneurs peut être traité de façon statique (l'optimisation concerne un nombre prédéfini de conteneurs) ou dynamique (tenant compte des changements dans les dates d'arrivée et de départ des conteneurs, de l'arrivée d'un nouveau navire, un bateau qui tombe en panne, etc...).

L'objectif de cette thèse est d'étudier chacune de ces quatre variantes du problème (problème statique à un seul type de conteneur, problème statique à plusieurs types de conteneurs, problème dynamique à un seul type de conteneur et problème dynamique à plusieurs types de conteneurs) et de proposer des solutions pour chacun d'entre elles.

Dans la littérature, le problème statique à un seul type de conteneur est le plus étudié. Des solutions ont été proposées par Preston et Kozan (2001), Chen et ses collègues (2004), Kim et Park (2003), Zhang et ses collègues (2003), etc. Contrairement au premier problème, la littérature relative aux trois autres variantes est rare et peu de travaux ont été proposés.

Cette thèse est composée de cinq chapitres. Le premier chapitre est consacré à la présentation d'un terminal. Les différentes opérations portuaires sont décrites et une étude bibliographique des différents problèmes relatifs à la gestion d'un terminal à conteneurs est réalisée. Dans le chapitre 2, les méthodes de résolution du PSC seront présentées et nous détaillons les 2 approches utilisées dans ce travail ; les algorithmes génétiques et la recherche harmonique. Le chapitre 3 est dédié au problème de stockage de conteneurs statique à un seul et à plusieurs types de conteneurs. Nous présentons la formulation mathématique du PSC pour les deux cas et nous détaillerons l'adaptation de l'algorithme génétique et de la recherche harmonique pour chaque variante du problème. Dans le chapitre 4, le problème dynamique à différents types de conteneurs est traité. Nous décrivons les deux approches proposées, basés sur les algorithmes génétiques et la recherche harmonique. Le chapitre 5 est consacré à la présentation des résultats de simulations des différentes approches de résolution du PSC statique et dynamique. Afin de vérifier la performance des approches proposées, une étude comparative des deux méthodes de résolutions utilisées pour chaque variante du problème est établie. Enfin, la conclusion générale présente un résumé de la thèse, des observations finales ainsi que des recommandations pour de futures recherches.

Chapitre 1 : Les terminaux à conteneurs et les opérations portuaires : état de l'art

Qui ne sait pas tirer les leçons de 3000 ans
vit seulement au jour le jour
[Goethe]

Introduction :

Avant la globalisation de la conteneurisation, les marchandises étaient conditionnées dans des cartons, des caisses marines ou des palettes. Cette méthode de manutention nécessite beaucoup de travail et est très coûteuse. L'apparition du processus de la conteneurisation a contribué à la réduction les frais de transport et à l'accroissement de la sécurité et de la qualité des marchandises manipulées.

La baisse des coûts de transport induit également l'accroissement de la prospérité au niveau du commerce international. Selon la CESAP (2005), la croissance du commerce mondial a été environ 1,5 fois la croissance économique mondiale entre 1950 et 1990. Le ratio a augmenté à 2,5 fois dans la période 1990-2004. Cette tendance indique que le commerce est devenu une composante de plus en plus cruciale de l'activité économique mondiale.

Le concept de la conteneurisation a été rapidement adopté par les industries du transport maritime et les ports maritimes ont migrés vers une nouvelle infrastructure nommée « terminal à conteneurs ».

I. Les terminaux à conteneurs

Pour répondre à la croissance du trafic conteneurisé et aux exigences imposées par les armateurs et le gigantisme des navires porte conteneurs, les ports maritimes ont adoptés une nouvelle infrastructure nommée « terminal à conteneurs ».

Un terminal à conteneurs est un endroit ou les conteneurs arrivant sur des navires sont déchargés par des grues à quai et transférés aux zones de stockage par des véhicules de levage dits cavaliers. Il forme ainsi le maillon essentiel dans la chaîne de transport maritime ; il remplit deux fonctions: le transfert et le stockage temporaire des conteneurs.

II. Présentation d'un conteneur

1. Définition

La Convention Internationale pour la Sécurité des Conteneurs (CSC) a normalisé en décembre 1972, la définition du conteneur :

« Le conteneur est un engin de transport de caractère permanent, et de ce fait assez résistant pour permettre un usage répété, spécialement conçu pour faciliter le transport des marchandises sans rupture de charge par un ou plusieurs modes de transport, conçu pour être assujetti et/ou manipulé facilement, des accessoires ayant été prévus à cet effet. »

Les conteneurs sont des boîtes généralement métalliques, leurs dimensions sont définies par la norme ISO (Organisation internationale de normalisation) de 20 pieds (6,058 m) ou 40

pieds (12,19m) de longueur, ils ont une hauteur de 8,6 pieds (2,591m) et une largeur de 8 pieds (2,438m). Ils sont destinés à contenir des marchandises et permettre leurs acheminements par différents modes de transport (route, rail, voies aérienne, fluviale et maritime).

2. Types de conteneurs

Il existe trois catégories de conteneurs ; les conteneurs pour marchandises générales, les conteneurs pour marchandises spécifiques et les conteneurs pour usage spécifique.

a. Les conteneurs d'usage général : appelés aussi conteneurs dry. Ils sont équipés de portes aux extrémités et destinés à des marchandises générales et sèches.

Figure 1.1. Conteneur dry

b. Les conteneurs pour usage spécifique :

- conteneur à toit ouvert (Open top) : la structure de ce type de conteneur est identique à celui du dry, mais, le toit est mobile et est généralement bâché pour empotage vertical (pièces volumineuses ou/et indivisibles).

Figure 1.2. Conteneur Open Top

- conteneur plate-forme (Flats) : Ils sont à parois latérales ouvertes. Il existe deux types de Flats ; des conteneurs à parois d'extrémités fixes et d'autres à parois d'extrémités mobiles. Les Flats sont les seuls à admettre, sous certaines conditions, des marchandises en dépassement de hauteur et/ou de largeur.

Figure 1.3. Conteneur plate-forme

c. Les conteneurs pour marchandises spécifiques :

Ils sont utilisés pour les marchandises ayant une caractéristique thermique spéciale. On peut citer :

- Conteneur ventilé : la surface de ventilation de ce type de conteneur est augmenté par l'ouverture d'orifices de ventilation dans les longerons. Il est utilisé pour le stockage de certains fruits et légumes, café en sacs nécessitant la circulation de l'air.

Figure 1.4. Conteneur ventilé

- Conteneur à température contrôlée muni d'un groupe générateur pouvant être branché sur le système électrique du porteur. On y distingue :
  - chauffé, maintient une température minimum
  - réfrigéré (reefer ou refrigerated), pour la conservation des produits alimentaires

Figure 1.5. Conteneur réfrigéré

- Conteneur à atmosphère contrôlée, pour ralentir ou accélérer le mûrissement des fruits ou légumes.
- Conteneurs citernes : Ces conteneurs sont répartis en 2 grandes familles :
  - Les citernes chimiques
  - Les citernes alimentairesLes conteneurs citernes sont utilisés donc pour des produits liquides, pulvérulents ou gazeux. Une citerne chimique ne peut pas contenir des produits alimentaires, alors qu'il est possible de transformer une citerne alimentaire pour la rendre chimique.

Figure 1.6. Conteneur citerne

- Conteneurs pour vrac : Ils ont une structure adoptée aux marchandises qui ne sont pas emballées ou arrimées (vrac) à savoir farine, grains. Ils sont équipés de trappes de chargement sur le toit, et de trappes de déchargement au bas d'une extrémité

III. Les opérations portuaires

Un terminal à conteneurs est un système complexe constitué d'un ensemble de processus logistiques reliés entre eux. C'est une séquence d'événements de l'arrivée d'un navire jusqu'au départ de containeurs ou vice-versa,

Les navires sont aujourd'hui débarqués et embarqués dans de grands terminaux. Le processus de chargement et de déchargement dans un terminal à conteneurs moderne typique peut être divisé en sous-processus (fig. 2.7)

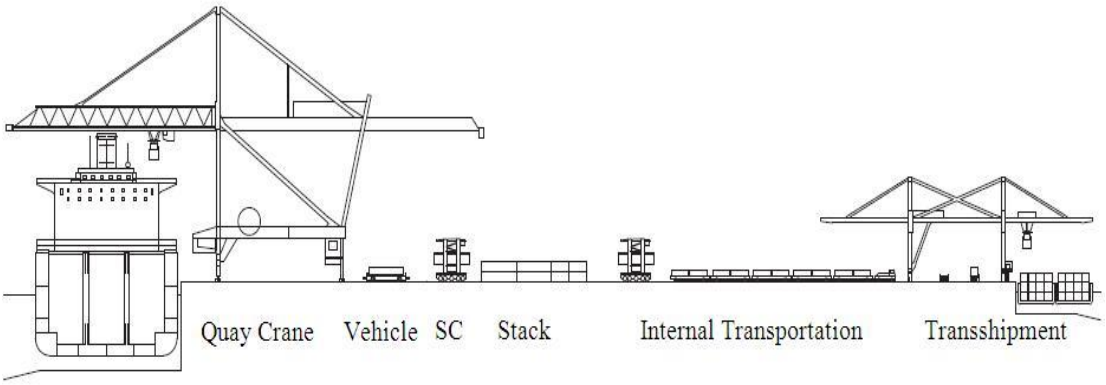


Figure 1.7. Le processus de chargement et déchargement dans un terminal (Vis et Koster, 2003)

Quand un navire arrive au port, les conteneurs d'importation doivent être retirés du navire et déposés sur la plate-forme en utilisant des grues de quai (Quay Crane). Ensuite, les conteneurs sont transférés par des véhicules aux piles ou zones de stockage (stack) où ils peuvent être stockés pendant une période.

Les conteneurs sont manipulés par des grues ou des chariots cavaliers (SC). Un chariot cavalier peut à la fois transporter et stocker un conteneur dans la pile.

D'autres équipements peuvent être utilisés pour le transport des conteneurs comme les véhicules à guidage automatique (AGV) et les grues d'empilement automatique. La figure ci-dessous présente les équipements utilisés dans un terminal.

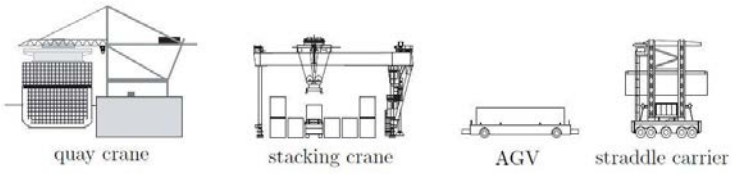


Figure 1.8. Équipements d'un terminal (Meersman, 2002)

Après une certaine période les conteneurs sont récupérés de la pile par des grues et transportés par des véhicules à un autre mode de transport (Trains, camions, autres navires).

Le processus de chargement et déchargement peut également être exécuté à l'envers, pour charger les conteneurs d'exportation sur un bateau.

Il existe plusieurs indices pour mesurer la performance d'un terminal à conteneurs. Le tableau ci-dessous les résume.

Tableau 1.1. Critères de performance (Ioannou et Jula, 2008)

Indices de performance	
Average Cycle time	Mesure la performance des opérations de chargement et déchargement. Il s'exprime en mouvements par heure
Throughput of crane	Nombre moyen de mouvements des grues par heure
Ship turnaround time	Temps (heure) pris par un navire pour effectuer l'opération du chargement/ déchargement
Truck turnaround time	Temps moyen qu'un camion reste dans le terminal. Ce temps ne prend pas en compte le temps passé par le camion dans la porte *(gate)*.
Gate utilisation	Pourcentage de temps pris pour servir le trafic entrant et sortant de conteneurs.
Container dwell time	Temps moyen qu'un conteneur reste dans le terminal avant d'être transporté
Idle rate of the equipment	Pourcentage de temps que l'équipement est inoccupé
Average cost per container	C'est le coût moyen par conteneur.

Les critères de performance les plus importants sont ceux qui reflètent la relation du terminal avec les clients. Généralement, les opérateurs portuaires visent à atteindre deux objectifs :

- Minimiser le temps moyen pris par les navires dans les postes d'amarrage (*Ship turnaround time),* qui est une mesure du niveau de service fourni par le terminal à ses clients, à savoir les compagnies maritimes.
- Maximiser le nombre moyen de mouvements des grues (*Throughput*), qui est une mesure de la productivité d'un terminal.

Afin d'atteindre des performances satisfaisantes, un certain nombre de décisions doit être pris au niveau opérationnel pour gérer les opérations ainsi que leurs répercussions. Par exemple, les décisions sur les emplacements de stockage des conteneurs affectent directement la répartition des grues et des conducteurs des grues.

En outre, la participation de plusieurs éléments dans les opérations et les interactions entre les différents processus de service complique le problème de gestion des opérations. Il est donc évident qu'il est très difficile d'atteindre des décisions optimales qui favorisent les objectifs globaux. L'une des solutions efficaces est la décomposition hiérarchique du problème initial en sous-problèmes plus simples.

La figure ci-dessous illustre une décomposition des processus dans un terminal à conteneurs.

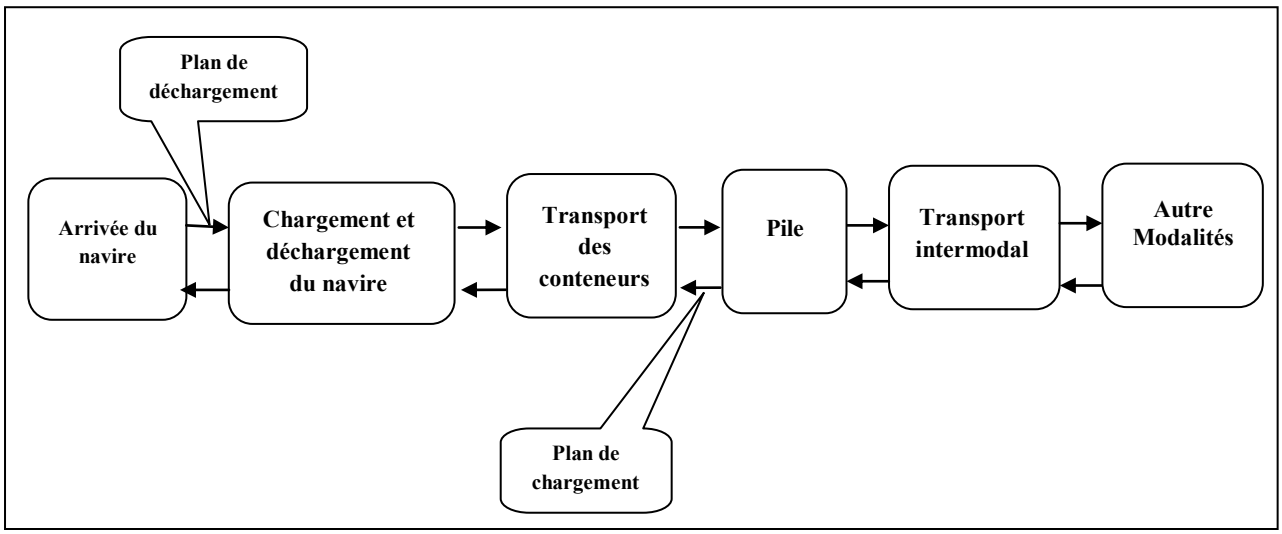


Figure 1.9. Les processus dans un terminal à conteneurs

Dans la section suivante, nous présentons une revue bibliographique des méthodologies utilisées pour résoudre les problèmes de décision rencontrés dans chaque processus.

IV. Revue bibliographique

1. Arrivée du navire

Quand un navire arrive au port, il doit être affecté à un poste d'amarrage (berth) pour s'amarrer au quai. La planification ou l'ordonnancement des postes d'amarrage consiste à assigner chaque navire à un poste disponible durant le temps prévu de sa permanence.

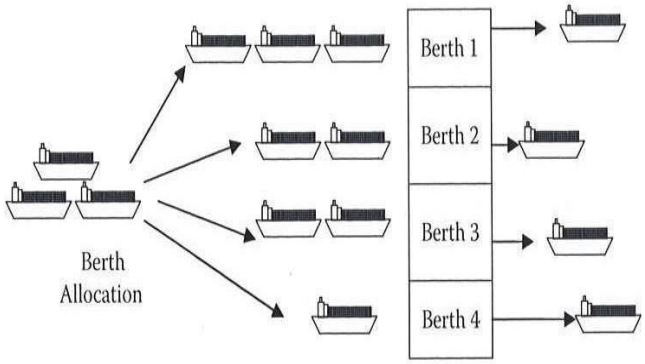


Figure 1.10. Ordonnancement des postes d'amarrage (Kim, 2008)

Le but d'une planification des postes d'amarrages est toujours de réaliser toutes les opérations d'un navire dans le temps accordé par l'opérateur du terminal et la compagnie de navires (Kim, 2008).

La durée de permanence d'un navire dépend du temps accordé au déchargement des conteneurs qu'il contient. Cette fonctionnalité est assurée par les grues de quai. En effet,

beaucoup de chercheurs traitent le problème de planification des postes et des grues de quai simultanément.

Imai et ses collègues ont beaucoup travaillé pour résoudre ce type de problème. En 1997, ils ont présenté un algorithme permettant l'allocation des postes d'amarrage aux navires tout en minimisant leurs temps de séjour et l'insatisfaction générée suite à un retard ou une avance dans leurs arrivées (Imai et col, 1997).

Kim et Moon (2003) proposent un modèle MIP (Mixed Integer Linear Problem) pour la planification des postes d'amarrage. L'objectif de ce travail est de réduire le coût généré par une gestion non optimale des postes d'accostage. Les résultats expérimentaux montrent que l'algorithme du recuit simulé fournit des solutions identiques aux meilleures solutions déterminées par le modèle MIP.

Park et Kim (2003) présentent un modèle de programmation en nombres entiers pour ordonnancer simultanément les grues du quai et les postes d'arrimage. La résolution du problème comprend deux phases. Le technique de sous gradient est utilisée en premier lieu pour l'assignation des postes d'accostage. Une méthode de programmation dynamique est appliquée par la suite à la solution optimale générée par la première phase pour l'ordonnancement des grues de quai.

Imai et col (2008) et Liang et col (2009) utilisent les techniques des algorithmes génétiques pour résoudre le même problème. L'objectif du l'équipe de Liang était de minimiser la somme des temps de manutention, d'attente, et de retard pour chaque navire.

2. Chargement et le déchargement du navire

Généralement, le nombre des conteneurs en import et qui doivent être déchargés dans le terminal n'est connu que peu de temps avant l'arrivée du navire. Le plan de déchargement indique les conteneurs à décharger dans chaque port et leurs emplacements sur le bateau.

Les conducteurs de grues de quai assurent l'opération de déchargement. Ils sont presque libres dans la détermination de l'ordre des conteneurs à décharger. En effet, une grande variance se produit au niveau des temps alloués à cette opération

Afin d'assurer un transbordement rapide et efficace des conteneurs, une bonne répartition des conteneurs sur le navire est nécessaire. Selon Shields (1984), les conteneurs chargés doivent satisfaire à un ensemble de contraintes liées essentiellement aux limites physiques du navire et des conteneurs et à l'ordre des ports destinataires.

Dans ce travail, la méthode de Monte Carlo est appliquée pour résoudre le problème de chargement des conteneurs. Le plan le plus efficace est généré. Il indique les emplacements exacts des conteneurs d'exportation dans le navire.

Selon Wilson et Roach (2000), la détermination d'une solution optimale pour le problème de chargement et déchargement des conteneurs semble difficile et même irréalisable dans des délais raisonnables. Il s'agit d'un problème complexe, qui dépend de la capacité du navire et du nombre de conteneurs déchargés et chargés à chaque port. Par conséquent, les auteurs proposent de décomposer le processus en deux sous-processus, à savoir un processus stratégique et un processus de planification tactique.

Dans la première phase, un algorithme Branch and Bound est appliqué pour affecter chaque conteneur à un bloc dans le navire. La recherche Tabou est utilisée par la suite pour assigner les conteneurs à des endroits spécifiques au sein des blocs déterminés dans la première phase. Ainsi, des solutions acceptables mais pas toujours optimales peuvent être déterminées en temps réel dans une durée de calcul raisonnable.

Dans les travaux de [Avriel et Penn, 1993] et [Avriel et col., 1998], un modèle mathématique pour la planification d'arrimage d'un porte-conteneurs est présenté sans prendre en considération la stabilité du navire. L'objectif est de minimiser le nombre de déplacement ou remaniement (shifting). En outre [Imai et col., 2002] a appliqué un modèle de programmation mathématique, mais ils ont fait de nombreuses hypothèses simplificatrices qui les rendent inappropriées pour des applications pratiques.

[Sciomachen et Tanfani, 2007] développent un algorithme heuristique pour résoudre ce même problème. Leur but est de réduire le temps de chargement total. Cette approche est comparée à une autre heuristique et les résultats ont montré son efficacité.

[Imai et col., 2006] proposent aussi un plan d'arrimage des conteneurs dans un navire qui répond à deux critères conflictuels, la stabilité du navire et le nombre minimal de remaniements inutiles des conteneurs. Le problème est formulé comme un programme multi-objectif en nombres entiers.

3. Transport de conteneurs d'un bateau à pile et vice versa

Comme décrit précédemment, les conteneurs doivent être transportés du navire aux emplacements de stockage (stack) et vice versa. La résolution de ce problème comprend deux parties. La première consiste à choisir le type de l'équipement de manutention qui prendra en charge le transport des conteneurs. La détermination du nombre nécessaire de véhicules participants dans cette opération fera l'objet de la deuxième partie.

Pour le transport d'un seul conteneur, des véhicules comme les chariots élévateurs, les camions ou les chariots cavaliers peuvent être utilisés.

Chariot à fourches

Camion

Chariot cavalier

Figure 1.11. Équipements de manutention

Pour le transport de plusieurs conteneurs, des systèmes multi-remorque peuvent être utilisés, voir Fig. 1.12.

Figure 1.12. Système multi-remorque

Dans les terminaux à conteneurs automatisés, des véhicules à guidage automatique sont utilisés pour le transport interne.

Figure 1.13. Véhicule à guidage automatique

Beaucoup de chercheurs ont traité le problème de transport des conteneurs à savoir [Bish et col, 2001], [Van der Meer, 2000], [Yu, 2010], etc...

L'objectif était souvent la minimisation la distance totale parcourue par les véhicules, la réduction du nombre de véhicules de manutention utilisés et de leurs temps de fonctionnement.

4. Le stockage (empilement) des conteneurs

L'opération du stockage est la partie la plus compliquée dans un terminal puisque les conteneurs en import et en export sont empilés simultanément dans un même espace de stockage.

Après l'arrivée d'un navire, les conteneurs importés sont déchargés et sont déplacés de la zone de rassemblement aux emplacements de stockage. Les conteneurs de la même longueur sont normalement empilés les uns sur les autres, et sont autorisés d'y rester pendant quelques jours gratuitement.

Le processus d'exportation est l'inverse du celui d'importation. Avant l'arrivée prévue du navire, le port accepte des conteneurs destinés à l'export. Généralement, leur arrivée est aléatoire. Ils sont stockés temporairement, dans une période qui dépend de la date de départ du navire. Avant le chargement des conteneurs dans le bateau, un plan de chargement est préparé en tenant compte de la destination finale, du type, du poids de chaque conteneur et du poids maximum autorisé pour chaque pile, ainsi que de la stabilité du navire.

On distingue deux manières de stockage de conteneurs; le stockage sur châssis ou l'empilage sur terrain. Avec le premier système (système de châssis), chaque conteneur est accessible de façon directe et individuelle. Alors qu'en cas d'empilage sur terrain, pour pouvoir accéder à un conteneur, il faut décharger tous ceux qui se trouvent sur lui. Vu la limite de l'espace alloué au stockage, l'utilisation du deuxième système est la plus courante.

La zone d'entreposage ou de stockage est un grand espace divisé en blocs (piles). Chaque bloc est composé de plusieurs baies chacune comprend aussi un ensemble de colonnes. La position d'un conteneur dans un block est identifiée par la baie, la colonne et l'étage (voir figure)

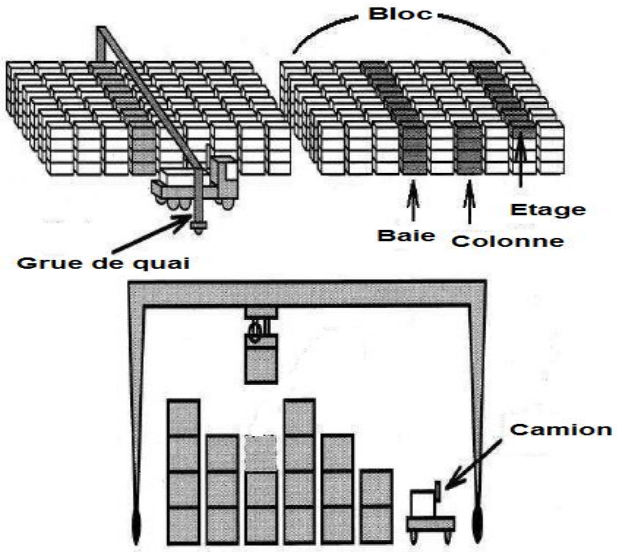


Figure 1.14. Espace de stockage

Il existe une distribution préliminaire des conteneurs dans les zones de stockage. Elle est basée sur divers critères afin de simplifier les opérations de transfert des conteneurs aux autres zones. Les conteneurs destinés à l'export se retrouvent généralement près de la zone d'opérations portuaires (près des grues de quai) afin de minimiser la distance parcourue par les véhicules de transport interne lors des opérations de chargement du navire.

Les conteneurs déchargés du navire et qui quitteront le terminal par transport ferroviaire, seront entreposés près des voies ferrées pour diminuer la distance à parcourir lors du chargement des trains. L'espace restant de la zone est utilisé pour le stockage des conteneurs vides et des conteneurs en import et qui quitteront le terminal par transport routier. [Dubreuil, 2008]

La figure ci-dessous présente une disposition d'un terminal à conteneurs.

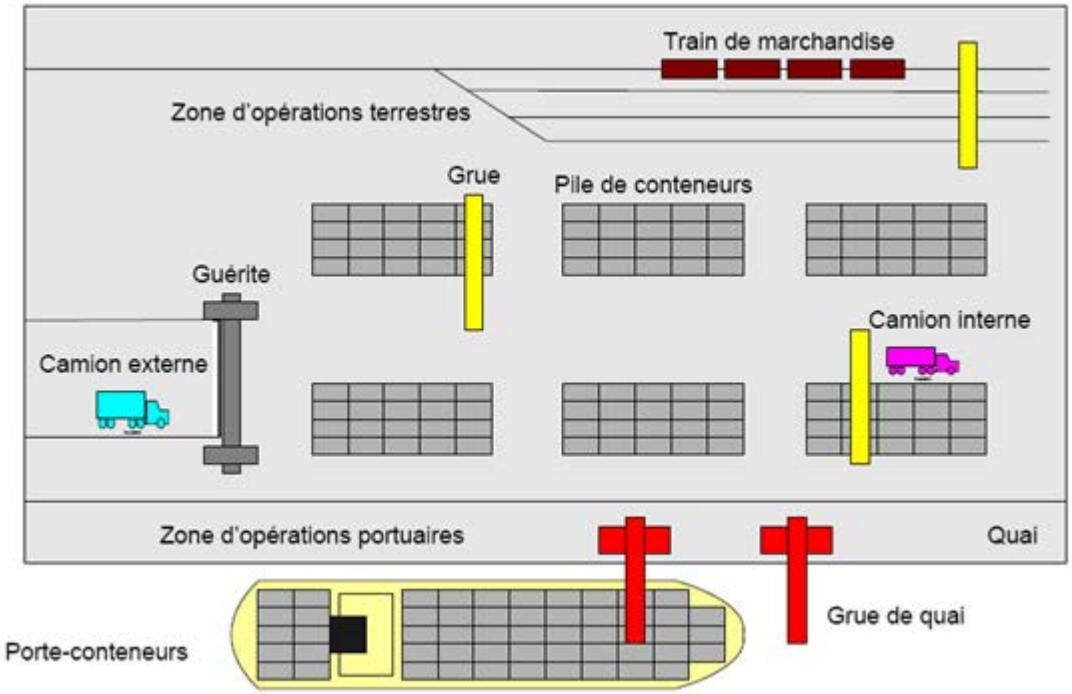


Figure 1.15. Structure d'un terminal à conteneurs

La superposition des conteneurs selon des critères et des stratégies étudiés assure une meilleure gestion de l'espace alloué au stockage mais ne peut pas éliminer toutes les opérations de remaniements.

Les mouvements de remaniements appelés aussi mouvements parasites (rehandling) consistent à déplacer un ou plusieurs conteneurs pour retirer le conteneur dont nous avons besoin lors du déchargement.

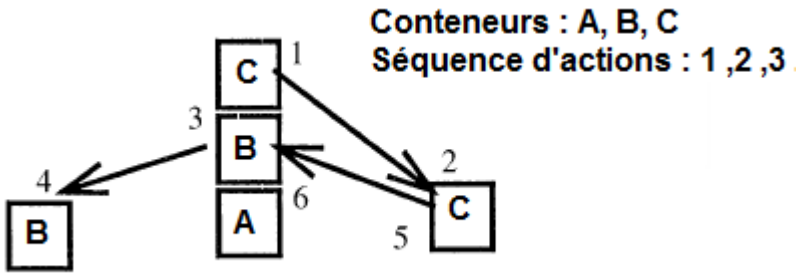


Figure 1.16. Récupération d'un conteneur au milieu de la pile

Dans l'exemple illustré dans la figure 1.16, l'extraction du conteneur B nécessite un remaniement du conteneur C.

Le stockage des conteneurs est parmi les décisions les plus importantes et les plus compliquées dans un terminal à conteneurs. Ainsi, l'affectation des tous les conteneurs

arrivant au port aux emplacements les plus adéquats favorise la minimisation des opérations de déplacements inutiles des conteneurs lors de leur transfert au navire, au camion ou au train. Ce qui minimise le temps de fonctionnement des grues de quai et le temps qu'un camion passe dans le terminal.

La complexité du problème de stockage des conteneurs réside au niveau de l'incertitude dans l'ordre de récupération des conteneurs. Le temps de départ des conteneurs en import est souvent incertain puisque les véhicules (les camions) assurant leurs transports arrivent plus au moins aléatoirement. Pour les conteneurs en export, leurs dates de départ est habituellement connu puisqu'il est relié à celui du navire. Néanmoins, le plan d'arrimage n'est prêt que peu de temps avant le chargement et parfois la date de départ du navire est retardée pour des raisons météorologiques.

Malgré la difficulté rencontrée dans la résolution du problème de stockage des conteneurs, plusieurs travaux l'ont traité et diverses approches ont été développées à savoir des systèmes d'aide à la décision, des modèles mathématiques et des méthodes heuristiques,…

De Castilho et Daganzo (1993), concluent que pour aboutir à une bonne configuration des blocs de stockage, il faut utiliser des méthodes pour estimer le nombre de mouvements nécessaires pour récupérer un conteneur en fonction de la hauteur de la pile et la stratégie de l'opération.

Kumar et Vlacic (2008) présentent un modèle analytique simple pour prévoir le temps nécessaire au déchargement des conteneurs et déterminer aussi le temps d'utilisation des équipements. Le modèle proposé est appliqué au port « Suva » et il a produit de bons résultats.

Preston et Kozan (2001) proposent un algorithme génétique pour résoudre le modèle de localisation des conteneurs en export dans un port. Leur objectif était de réduire le temps de transfert et le temps de manutention des conteneurs et par la suite le temps passé par les navires à quai. Cette approche est appliquée dans le port de Brisbane et elle a généré de bons résultats surtout en la comparant avec le processus déjà utilisé dans ce port.

Chen et ses collègues (2004) ont combinés plusieurs métaheuristiques (recherche Tabou, recuit simulé et algorithme génétique) pour résoudre le problème d'optimisation des zones de stockage dans un port. Le problème est défini comme un problème de Bin-Packing à deux dimensions et qui vise à réduire l'espace alloué au chargement dans un intervalle de temps.

Kim et Kim (2007) ont présenté quelques méthodes pour déterminer le coût optimal pour l'opération du stockage des conteneurs en import. La contribution de ce travail est que le

modèle du coût combine les frais de manutention pour les clients (frais de stockage) et les exploitants de terminaux.

Lee et col. (2009) ont développé un algorithme heuristique pour résoudre l'ordonnancement des camions internes (dans le port) et le problème d'allocation de stockage. Leur objectif est de minimiser la somme pondérée du total des retards à une demande et le coût des tournées des camions.

Dans Chen et col. (2000), le problème d'allocation d'espace de stockage est examiné. Un réseau espace-temps est développé pour aider à affecter à l'avance les conteneurs à des emplacements de stockage. Ce réseau peut représenter des entités en mouvement dans le temps et l'espace. Par la suite, un modèle de programmation mathématique peut être développé. L'objectif de cette méthode est de minimiser le coût total de l'opération. Un test est effectué pour un cas réel en utilisant l'algorithme Branch & Bound.

Kim et Park (2003) ont proposé une règle de décision heuristique et une technique d'optimisation de type sous-gradient pour résoudre l'allocation d'espace de stockage pour les conteneurs en export. Leur objectif était de trouver un arrangement efficace des conteneurs dans l'espace de stockage qui optimise l'opération de chargement. Une étude expérimentale a été réalisée pour comparer ces deux algorithmes.

Kim (1997) a présenté une méthode d'estimation du nombre total des remaniements pour retirer tous les conteneurs en import dans une baie. Cette méthode consiste en un ensemble de tables et d'équations qui ont examiné les diverses configurations possibles des empilements de conteneurs.

Kim et Kim (1998) ont proposé un modèle de coût qui estime les coûts des différents équipements qui interviennent dans l'opération de manutention des conteneurs en import et détermine par la suite l'espace de stockage et le nombre de grues optimal.

Zhang et ses collègues (2003) ont résolu le problème d'allocation d'espace de stockage en utilisant l'approche « Roulling-Horizon ». Ils ont considéré deux types de conteneurs, en import et en export. Leur objectif était de réduire la distance de transport total de conteneurs entre les blocs de stockage et les postes d'amarrage des navires. Pour chaque horizon de planification, ils décomposent le problème en deux niveaux et chacun est formulé comme un modèle de programmation mathématique.

Bazzazi et col. (2009) ont étendu le problème d'allocation d'espace de stockage proposé dans la littérature (Zhang et col., 2003). Ils ont considéré des conteneurs de différents types (réfrigéré, vide, dry) et de différentes tailles. Les auteurs ont proposé un algorithme génétique

pour résoudre ce problème et ils ont supposé que les blocs alloués pour chaque type de conteneur sont connus à l'avance.

5. Le transport inter-terminal et les autres modes de transport

Les conteneurs doivent être récupérés de la zone de stockage pour être transportés par différents modes de transport.

Les conteneurs destinés à l'export sont stockés dans le port alors que ceux en import doivent être transférés pour être transportés par un navire ou par voie ferroviaire ou routière. Afin d'optimiser l'exploitation des différentes ressources et d'améliorer la qualité du service clientèle, une meilleure gestion des opérations relatives au transport intermodal est nécessaire. Ces opérations concernent la gestion de la flotte de véhicules, l'affectation des ressources, le routage des véhicules (VRP) pour les activités de livraison et de ramassage, etc.

Le problème de routage des véhicules consiste à déterminer les routes permettant de visiter tous les clients tout en minimisant le coût du transport, en satisfaisant les demandes de ces derniers et en respectant les différentes capacités des véhicules.

Kammarti et ses collègues se sont intéressés à une importante variante du VRP qui est le problème de collecte et distribution à fenêtres de temps avec un seul véhicule. Ils ont utilisé les algorithmes évolutionnistes et la recherche Tabou comme outils d'optimisation. (Kammarti et col. 2004), (Kammarti et col. 2005a), (Kammarti et col. 2005b)

Macharis et Bontekoning (2004) ont présenté une étude bibliographique concernant la définition des problèmes rencontrés dans le transport intermodal ainsi que des méthodes récentes développées pour les résoudre.

Kozan (2000) a proposé un modèle réseau pour l'étude des opérations de transfert des conteneurs dans un terminal multimodal. Le modèle proposé peut être utilisé comme un système d'aide à la décision.

Kozan et Casey (2006) ont élaboré un modèle permettant de minimiser les retards des navires dans les terminaux à conteneurs. Des problèmes de grandes tailles ont été résolus en utilisant différentes métaheuristiques. Corry et Kozan (2006) ont étudié la planification du chargement des trains intermodaux. Ils se sont intéressés à la double manutention de conteneurs et à la planification des trains. Un modèle pour la génération d'un plan de chargement a été développé pour un environnement dynamique.

V. Problème de Stockage de Conteneurs (PSC)

Dans le but de maximiser la performance d'un port, un ensemble de décisions doit être pris lors de la gestion d'un terminal à conteneurs. Ces décisions se résument essentiellement dans

l'allocation et l'ordonnancement d'un ensemble de ressources telles que les postes d'amarrage, les véhicules de manutention (grues de quai, chariot à fourches, chariot cavalier, véhicule à guidage automatique,..), l'espace de stockage.

Le nombre moyen de mouvements des grues par heure (Throughout time) et le temps moyen qu'un camion reste dans le terminal (Truck Turnaround time) sont des critères de performance couteux. Ainsi chaque port cherche à les minimiser en optimisant les processus de chargement et déchargement qui représentent sa partie majeure.

Le présent travail a pour objectif de déterminer un plan de stockage des conteneurs destinés à l'import et à l'export, dans les emplacements disponibles dans les zones de stockage du port. Le plan établi doit assurer un chargement efficace des conteneurs lors de leur transfert vers un navire, un camion, ou un train.

Le problème de stockage des conteneurs ou Container Stacking Problem (PSC) peut être définit comme un problème d'affectation des conteneurs de différents types et de différentes tailles arrivant à un port, à des emplacements vides dans les blocks de stockage se trouvant dans le port.

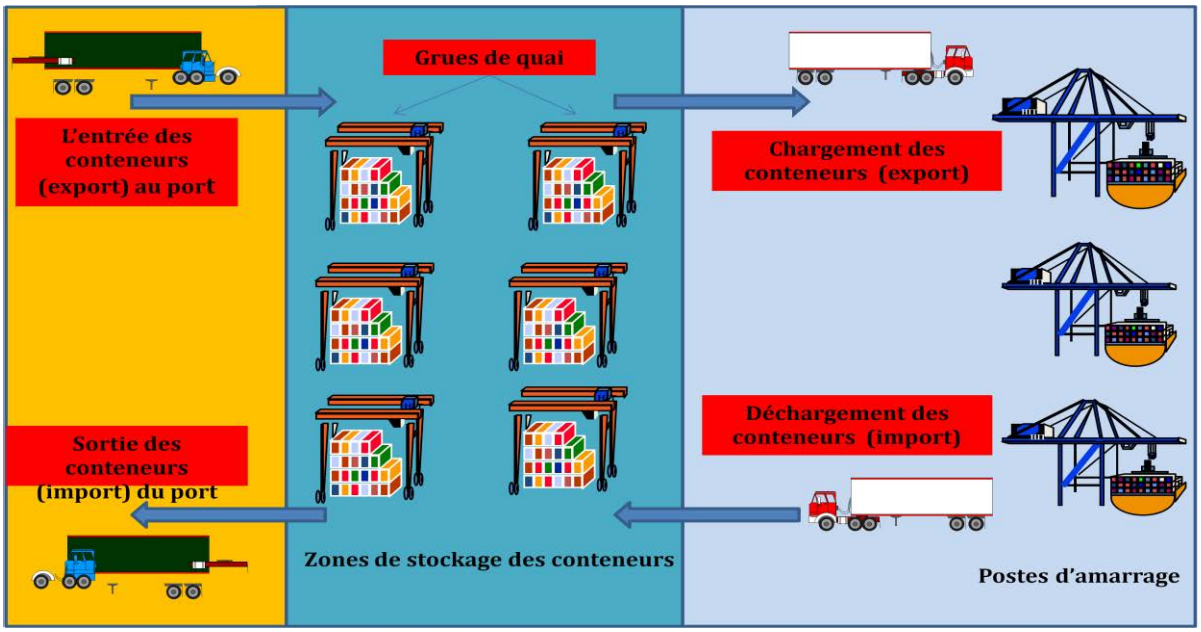


Figure 1.17. Les processus de chargement et déchargement dans un terminal à conteneurs

Le PSC peut être formulé comme un problème de Bin-Packing à 3 dimensions où les conteneurs représentent les objets et les zones de stockage les boîtes utilisées pour le rangement. Il appartient à la catégorie des problèmes NP-difficiles et NP-complets.

A chaque port de destination, des conteneurs sont déchargés du navire et chargés dans le port pour être livrés à leurs clients. En fait, il faut trouver presque en temps réel le meilleur

emplacement pour un conteneur sur le port de manière à minimiser le nombre de conteneurs manipulés lors de son extraction au moment de son départ pour être chargé sur le navire, train ou camion associé.

La majorité des travaux qui ont traité ce problème a utilisé des modèles mathématiques et stochastiques. Toutefois, ces approches ne peuvent pas être appliquées pour des problèmes complexes et de grandes tailles. En plus, ils ne considèrent pas l'aspect dynamique du problème.

Pour remédier à ces limites, il est plus avantageux d'adapter des métaheuristiques qui permettent de générer de bonnes solutions en un temps tolérable.

Conclusion

Dans ce chapitre, nous avons présenté les différentes opérations et problèmes en logistique portuaire. Dans la partie état de l'art, nous avons énuméré les différentes techniques qui ont été utilisées pour résoudre les problèmes de la planification des postes d'amarrage, de chargement et déchargement des navires, de transport de conteneurs du bateau à la zone de stockage et vice versa, de stockage des conteneurs (PSC) et de transport inter-terminal et les autres modes de transport.

Nous avons détaillé l'étude du problème de stockage de conteneurs (PSC) qui correspond au problème traité dans cette thèse.

Dans le prochain chapitre, nous présentons les méthodes de résolution utilisées et nous détaillerons l'algorithme génétique et la recherche harmonique, les approches adoptées pour la résolution du PSC.

Chapitre 2 : Les méthodes de résolution du PSC, les algorithmes génétiques et la recherche harmonique

Dans les sciences, l'intuition du dilettante peut avoir
une portee parfaitement identique à celle du specialiste,
et même parfois plus grande.
[Max Webber]

Introduction :

Pour résoudre les problèmes d'optimisation, deux familles d'approches peuvent être utilisées ; les méthodes exactes et les métaheuristiques.

L'avantage d'une méthode exacte est qu'elle fournit le résultat optimal pour le problème. Parmi ces méthodes, on peut citer les techniques de séparation et évaluation (branch-and-bound), les algorithmes avec retour arrière (backtracking), etc

Toutefois, c'est parfois impossible d'atteindre la solution optimale avec des problèmes de grandes tailles en un temps raisonnable, c'est le cas des problèmes NP-complets et des problèmes NP-difficiles. Ces techniques sont lentes et nécessitent un temps de calcul important.

Pour faire face à ces problèmes la majorité des travaux est concentrée sur les métaheuristiques qui permettent de générer, en un temps raisonnable, de bons résultats sans toutefois garantir leur optimalité.

Les métaheuristiques sont basées sur des simulations pour résoudre des problèmes d'optimisation complexes. Le facteur commun entre les métaheuristiques est qu'elles combinent les règles et l'aspect aléatoire pour imiter les phénomènes naturels. Ces phénomènes comprennent les processus d'évolution biologique. L'algorithme génétique (AG) se fonde sur des concepts de la génétique naturelle darwinienne. La recherche tabou proposée par [Glover, 1977] s'inspire du comportement des animaux et le recuit simulé proposé par [Kirkpatrick and al., 1983] traduit le processus de recuit physique.

Au cours de la dernière décennie, ces métaheuristiques ont été utilisées pour surmonter les carences de plusieurs méthodes numériques classiques. Toutefois, des nouvelles heuristiques plus puissantes basées sur des analogies avec des phénomènes naturels ou artificiels, doivent être explorées. A titre d'exemple, nous pouvons citer les algorithmes de colonie d'abeilles simulée, la méthode de l'essaim particulaire et la recherche harmonique qui imite le comportement d'un musicien qui improvise afin de trouver la meilleure harmonie.

Au cours de ce chapitre, nous commençons par présenter les méthodes de résolution utilisées pour résoudre le problème de stockage des conteneurs. Ensuite, nous présentons les algorithmes génétiques et la recherche harmonique, les deux métaheuristiques utilisées pour la résolution des problèmes traités. Ce choix est dû à diverses raisons. Le succès des méthodes basées sur les AG n'a cessé de grandir depuis leur apparition, notamment dans le domaine de la recherche opérationnelle et de l'intelligence artificielle et surtout pour la résolution des

problèmes d'ordonnancement. En ce qui concerne la recherche harmonique, elle a été adoptée pour divers problèmes comme le problème de routage de véhicules et a enregistré de bons résultats.

Ce chapitre est composé de deux parties. Dans la première, les différentes étapes d'un algorithme génétique seront détaillées. Et la deuxième partie sera dédiée à la description de la recherche harmonique.

I. Les méthodes de résolution de PSC

Le problème de stockage de conteneurs consiste en la détermination d'un arrangement d'un ensemble de conteneurs dans les différents blocs de stockage disponibles dans un port. Ce problème, qui est NP-difficile et NP-complet, est largement étudié dans la littérature et plusieurs approches ont été proposées pour résoudre ces différentes variantes. Parmi lesquelles il y a des méthodes exactes et d'autres sont des heuristiques. Dans cette section, un nombre de ces techniques va être décrit et leur adaptation au PSC sera présentée.

1. Les méthodes exactes :

Une méthode exacte est une approche qui permet de générer la solution optimale d'un problème. Son principe consiste généralement en une énumération exhaustive de l'ensemble de solutions possibles. De ce fait, il sera difficile, voir impossible, de trouver la solution optimale si la taille du problème devient grande. En plus, ces techniques sont très lentes et nécessitent une durée d'exécution très grande. Toutefois, il existe certaines méthodes plus intelligentes qui permettent de résoudre des problèmes de tailles plus ou moins larges, en minimisant le nombre de ces solutions réalisables. Parmi elles, on peut citer l'algorithme de branch and bound, branch and cut ou la programmation dynamique.

a. Branch and Bound :

C'est une méthode de la recherche arborescente ou de la procédure de séparation et d'évaluation, cette méthode consiste en la construction d'un arbre de recherche qui sera exploré de manière à éviter les branches inutiles qui sont des branches contenant des solutions non intéressantes ou carrément non réalisables. L'exploration se fait avec des évaluations des branches et des comparaisons avec une valeur seuil du critère à optimiser. Cette technique donne de bons résultats pour les problèmes d'ordonnancement de petites tailles, mais dans le cas contraire, elle risque de générer des branches très étendues. Cette méthode a été utilisée dans [Gendron et Crainic, 1995] pour la résolution du problème de planification du transport terrestre de conteneurs.

b. Branch and cut :

Elle est aussi appelée méthode de programmation en nombres entiers. Comme toute méthode énumérative implicite, l'algorithme construit une arborescence nommée *l'arbre du "Branch and Cut"*, les sous-problèmes qui forment l'arbre sont appelés *des noeuds*. Il existe trois types de noeuds dans l'arbre du "Branch and Cut", le noeud courant qui est en train d'être traité, les noeuds actifs qui sont dans la liste d'attente des problèmes et les noeuds inactifs qui ont été élagués au cours du déroulement de l'algorithme. Le principe est de partir d'une solution admissible entière du problème, et à l'aide du simplexe par exemple, d'aller vers une autre solution admissible entière jusqu'à l'optimum.

c. Programmation dynamique :

Cette méthode se base sur le principe de Bellman : «Si C est un point qui appartient au chemin optimal entre A et B, alors la portion de ce même chemin allant de A à C est le chemin optimal entre A et C. » [Borne et al 1990]. Pour obtenir le chemin optimal du problème, il suffit donc de construire les différents sous chemins optimaux en partant de l'objectif et en remontant vers l'origine.

Beaucoup de chercheurs ont utilisés la programmation dynamique pour résoudre le PSC. [Kim et Bae, 1998], [Korbaa et Yim, 2004]et [Kim et col., 2000].

2. Les méta-heuristiques :

Une méta-heuristique est une approche efficace qui permet de traiter une large variété de problème. Elle permet de générer de bonnes solutions en une durée raisonnable sans garantir leur optimalité ou même leur réalisabilité. Ces approches peuvent être classées en deux familles, les méta-heuristiques de recherche locale et les méta-heuristiques évolutionnaires.

a. Les méta-heuristiques de recherche locale :

Le processus d'optimisation pour ces approches commence par une solution initiale. Par la suite, la procédure effectue des recherches de nouvelles solutions dans le voisinage de la solution courante, selon une fonction v(x), appelé généralement fonction de voisinage. Dans le PSC, la fonction de voisinage peut être la permutation des positions de deux ou de plusieurs conteneurs dans l'espace de stockage. Parmi les méta-heuristiques de recherche locale ou de voisinage, on peut citer la méthode de descente, la recherche Tabou, le recuit simulé, etc.

a.1. La recherche Tabou :

La recherche Tabou a initialement était proposée en 1986 par Glover [Glover 1986] et a connu un grand succès grâce aux résultats qu'elle a fournit pour de nombreux problèmes.

L'apport principal de cette approche vient du fait de l'utilisation d'une mémoire (appelé mémoire Tabou ou liste Tabou) afin d'éviter les problèmes de cycle. En effet, L'idée de base de la liste taboue consiste à mémoriser les configurations ou régions visitées et à introduire des mécanismes permettant d'interdire à la recherche de retourner trop rapidement vers ces configurations. D'autre part, cette approche permet d'éviter le blocage dans un extremum local. Ceci est du au fait qu'à chaque itération, l'algorithme tabou choisit le meilleur voisin non tabou, même si celui-ci dégrade la fonction de coût. Pour cette raison, on dit de la recherche avec tabou qu'elle est une méthode agressive.

L'algorithme de recherche Tabou, décrit par la figure 1, est composé de deux étapes. Dans la première (étape d'initialisation), l'algorithme commence par la génération d'une solution initiale, généralement de façon aléatoire, qui va être considérée comme la meilleure solution. En plus, il y aura une initialisation de la liste taboue. Par la suite dans la deuxième, une exploration du voisinage de la solution courante, les solutions taboues exceptées, sera établie. La meilleure solution trouvée par ce processus sera la solution courante pour l'itération suivante. En plus, cette solution sera gardée si celle-ci est meilleure que toutes les autres déjà trouvées par l'algorithme. L'étape 2 sera répétée jusqu'à vérification de la condition d'arrêt qui peut être un nombre maximum d'itérations ou un nombre maximum d'itérations sans amélioration de la meilleure solution.

```
Algorithme RechercheTabou
Etape 1 : initialisation
        F(x) ← Génère solution initiale
        Liste tabou = vide
        x* = x
        x_min = x
Etape 2 :
        Répéter
            x ← la meilleure solution du voisinage de F(x*) excepté voisins dans Liste Tabou
            Si f(x) < f(x_min) Alors
                x_min ← x
            Fin si
            Mise A Jour (Liste Tabou)
            x* = x
        Jusqu'à condition d'arrêt
```

Figure 2.1. Algorithme de la recherche Tabou

Toutefois, dans certains cas, les interdictions occasionnées par la liste tabou peuvent être jugées trop radicales. En effet, on risque d'éliminer (en les rendant tabous), certains mouvements particulièrement utiles. Autrement dit, il s'agit d'assouplir le mécanisme de liste

taboue, via un critère d'aspiration, tout en faisant attention de ne pas s'introduire dans un cycle. Par exemple, un critère d'aspiration rudimentaire peut consister à accepter un mouvement s'il conduit à une configuration meilleure que la meilleure configuration déjà trouvée.

Les travaux de [Wilson et Raoach, 2000], [Chen et col., 2007] ont utilisé la recherche Tabou pour résoudre le problème de manipulation (Handling) des conteneurs et ces approches ont fournit des bons résultats.

a.2. Le recuit simulé :

Le recuit simulé est un algorithme modélisé en analogie avec le phénomène du recuit qui peut être décrit par le chauffage d'un produit métallurgique à une température suffisante pour assurer son équilibre physico-chimique et structural, et que l'on fait suivre d'un refroidissement lent. De façon analogue, pour les l'algorithme SA, une solution initiale est sélectionnée. Par la suite, une exploration de son voisinage sera établie et une nouvelle solution sera choisie. Si elle est meilleure que la solution courante, elle prendra sa place, sinon elle sera sélectionnée avec une probabilité p qui dépend d'un paramètre, appelé température, qui sera réduite à son tour. Cette probabilité diminue de façon exponentielle, soit avec le temps soit suite à la dégradation de la solution courante. L'algorithme de recuit simulé peut être décrit par la figure suivante :

Algorithme **Recuit simulé**
Etape 1 : initialisation
F(x) ← Génère solution initiale
Sélectionner une température initiale $T > 0$
$x^* = x$
$x_{min} = x$
Etape 2 :
Répéter
$x \leftarrow$ une solution du voisinage de $F(x^*)$
$\Delta = F(x) - F(x_{min})$
Si $\Delta \leq 0$ Alors
$x_{min} \leftarrow x$
$x^* \leftarrow x$
Sinon
$x^* \leftarrow x$ avec une probabilité $P = e^{-\Delta/T}$
Fin si
Mettre à jour la température
Jusqu'à condition d'arrêt

Figure 2.2. Algorithme Recuit Simulé

Dans cet algorithme, la solution à sélectionner du voisinage de la solution courante peut se faire de façon aléatoire, selon l'approche Best fit (meilleur solution dans son voisinage) ou

First fit (la première solution meilleure que la solution courante, si elle existe). De plus, le recuit simulé permet la diversification vu qu'il accepte une solution de qualité inférieure avec une probabilité *P* qui dépend du degré de dégradation de la solution ainsi que du facteur température, *T*. Concernant la valeur de la température T, elle sera initialement élevée ce qui augmentera la probabilité d'acceptation des solutions de qualité inférieure et augmentera le degré de diversification. Par la suite, T diminuera, selon un facteur α à définir, lentement au fur et à mesure du déroulement de l'algorithme pour simuler le processus de refroidissement des matériaux. Sa diminution est suffisamment lente pour que l'équilibre thermodynamique soit maintenu. Enfin, le critère d'arrêt de l'algorithme peut être un nombre maximal d'itération, une valeur minimale de la température finale ou un nombre d'itérations sans amélioration de la meilleure solution. [Kang et col., 2006] [Chen et col., 2004]

Il manque le palier de température à l'algorithme

b. Les méta-heuristiques évolutionnaires :

Contrairement aux méthodes de recherche locale où une solution est considérée, dans les méat-heuristiques évolutionnaire toute une population de solutions contribue au processus d'optimisation. De ce fait, une telle méthode peut générer plusieurs solutions efficaces en parallèles via une auto-adaptation de chacune des solutions, d'une part, et la coopération des différentes solutions de la population, d'autre part. Parmi les méta-heuristiques évolutionnaires, on peut citer principalement, l'algorithme génétique, les colonies de fourmis, la recherche harmonique, etc.

b.1. Les colonies de fourmis

L'algorithme de colonie de fourmis a était été proposé en 1991 par Colorni, Dorigo et Maniezzo pour résoudre le problème du Voyageur de commerce. Elle a été appliquée avec succès à d'autres problèmes d'optimisation combinatoire. Cette approche imite le comportement des fourmis à la recherche de la nourriture. Dans la réalité, les fourmis communiquent grâce à une substance chimique, appelé phéromone, qui les aide à trouver le plus court chemin entre leur nid et les sources de nourriture. Ce processus peut être décrit comme suit : initialement, les fourmis quittent leur nid vers une source de nourriture et lorsqu'elles sont confrontées à un obstacle, un sous-ensemble de fourmis (en moyenne la moitié) choisit au hasard le chemin le plus court, tandis que les autres choisiront le plus long. Comme les fourmis se déplacent avec à peu près la même vitesse, le premier sous-ensemble sera de retour au nid plus rapidement et donc marquera son trajet deux fois, avant le retour d'autres fourmis, ce qui renforce la quantité de phéromone sur le chemin plus court. Comme

les fourmis empruntent provisoirement le chemin ayant la plus grande trace de phéromone, la probabilité de sélection des chemins plus courts par les fourmis suivantes sera plus grande. Ainsi, au fil du temps, comme le nombre de fourmis qui parcourent le chemin le plus court augmente, la quantité de phéromones s'accumule rapidement sur eux et en contre partie les plus longs chemins sont moins renforcés. De ce fait, les chemins indésirables seront moins visités et les fourmis finiront par l'adoption du chemin le plus court.

Par analogie au phénomène réel, l'algorithme de colonie de fourmis qui est un algorithme constructif, construit une solution étape par étape en se basant sur les informations relatives aux dépôts phéromones. Pour cela, on représente une solution sous forme d'un graphe est à chaque étape une arête est choisie selon plusieurs critères. Pour imiter le comportement des fourmis, l'algorithme se base sur trois règles principales. La première, la règle de transition, est utilisée pour identifier la prochaine étape dans le processus de génération d'une nouvelle solution. Cette règle repose principalement sur les quantités de phéromones déposées et l'arête la plus visitée sera généralement choisie. Toutefois et afin de garantir une diversification des solutions, une arête sera choisie aléatoirement avec une probabilité, généralement minime, à définir. La deuxième règle, appelée règle de mise à jour locale permet de modifier la quantité de phéromone sur chaque arête visitée. Et enfin, la règle de mise à jour globale permet de modifier la quantité de phéromone des arêtes représentant la meilleure solution trouvée. D'autre part elle minimise cette quantité pour les autres arêtes afin de simuler l'aspect d'évaporation de la phéromone.

```
Algorithme Colonie de fourmis
Etape 1 : initialisation
        Initialiser la quantité de phéromone de chaque arête à une valeur minimale
Etape 2 :
        Répéter
            Affecter, aléatoirement chaque fourmi à un nœud de départ
            Répéter
                Pour chaque fourmi faire
                   Début
                        Appliquer la règle de transition pour sélectionner le prochain nœud
                        Appliquer la règle de mise à jour locale
                    Fin
            Jusqu'à ce que chaque fourmi construise une solution
            Appliquer la règle de mise à jour globale
        Jusqu'à condition d'arrêt
```

Figure 2.3. Algorithme Colonie de fourmis

Cette méta-heuristique a été adopté pour le problème de chargement des conteneurs dans les travaux de [Liang et col., 2007] et [Lee et col., 2005].L'algorithme de colonie de fourmis peut être décrit par la figure 2.3

b.2. Les systèmes multi-agents (SMA) :

En 1999, Weiss a définit l'agent comme " une entité computationnelle " ou un robot, qui peut être vu comme percevant et agissant de façon autonome sur son environnement. On peut parler d'autonomie parce que son comportement dépend au moins partiellement de son expérience.

Il existe en fait plusieurs types d'agents, qui selon les capacités qu'ils possèdent, seront qualifiés de réactifs, cognitifs ou hybrides.

Un système multi-agent est une méthode de modélisation, simulation et résolution distribuée. C'est un ensemble homogène ou hétérogène d'agents situés dans un même environnement et qui interagissent.

Les types courants d'interaction dans un SMA incluent la coopération (travailler ensemble à la résolution d'un but commun) ; la coordination (organiser la résolution d'un problème de telle sorte que les interactions nuisibles soient évitées ou que les interactions bénéfiques soient exploitées) et la négociation (parvenir à un accord acceptable pour toutes les parties concernées). [Kefi, 2008].

Les SMA ont eu un grand succès dans la résolution du problème de PSC, parmi ses travaux nous citons [Henesey, 2004], [Kefi et col., 2007].

II. Les algorithmes génétiques :

1. Description :

Dans l'évolution naturelle, une combinaison de la structure génétique des parents génère un ensemble d'individus. Les opérations de croisement et de mutation subies par les individus enfants agissent sur leurs structures. Ainsi, le croisement assure le respect du phénomène d'hérédité et la mutation permet d'éviter une convergence prématurée de l'algorithme.

Les algorithmes génétiques ont été introduits en 1975 par John Holland. Ce sont des procédures de recherche inspirées des principes de la sélection naturelle et de la génétique. Ils appartiennent à la classe des algorithmes évolutionnaires qui permettent de modéliser le processus d'apprentissage d'une population d'individus pour s'adapter aux différents types de problèmes d'ordonnancement. [Kammarti R., 2006]

C'est une technique puissante pour résoudre les problèmes d'optimisation et trouver rapidement une solution quasi-optimale.

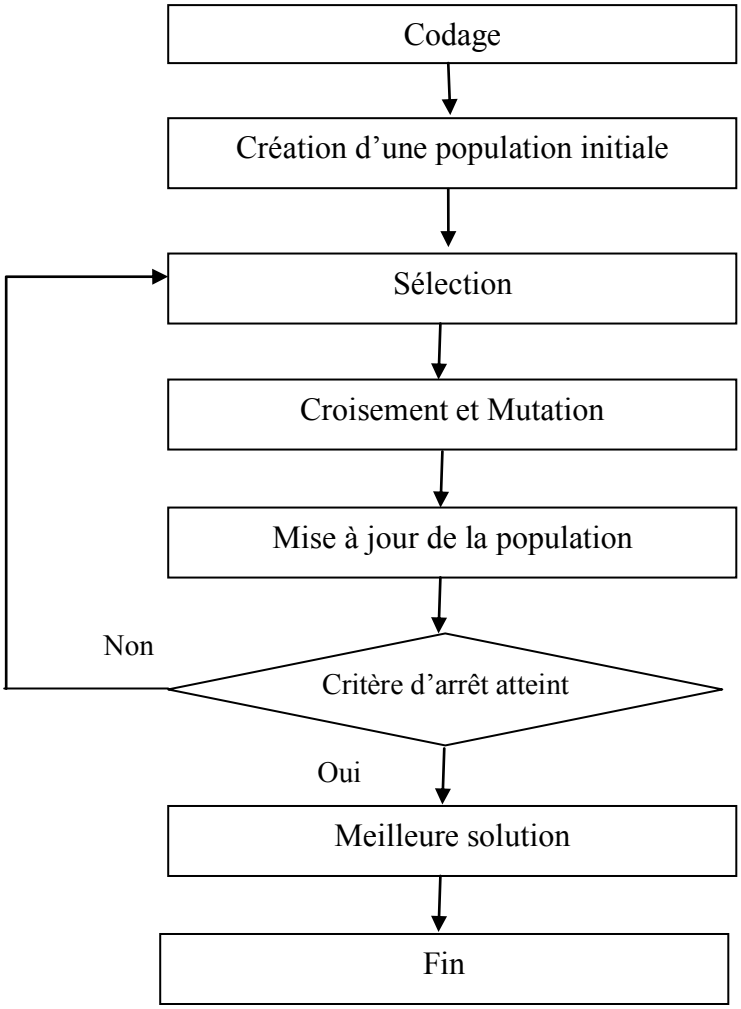


Figure 2.4. Le principe des algorithmes génétiques

Son fonctionnement est assez simple, il suit une démarche analogique à celle de l'évolution naturelle. La première étape consiste à créer un ensemble d'individus ou chromosomes générés aléatoirement pour former la population initiale.

Chaque individu représente une solution particulière au problème à résoudre. Les solutions candidates sont évaluées pour déterminer la performance de chacune. La phase suivante comprend une sélection aléatoire de deux parents de la population initiale. Les paires choisies subissent en premier lieu une opération de croisement suivi éventuellement d'une opération de mutation. Les opérations génétiques sont appliquées selon des probabilités de croisement

et de mutation fixées par l'utilisateur. La création des nouveaux individus se répète jusqu'à l'obtention d'une solution ayant la performance souhaitée. La figure ci-dessus décrit brièvement le fonctionnement d'un algorithme génétique.

2. Le codage

Le codage est la première étape dans la résolution d'un problème utilisant les algorithmes génétiques. Comme dans la biologie, chaque individu de la population est codé par un chromosome [Holland, 1975]. En AG les solutions sont codées par une structure de données contenant l'ensemble des variables de décision liées au problème.

Ainsi, le codage est une représentation contenant toute l'information nécessaire à la description d'une solution (chromosome) dans l'espace de recherche.

Beaucoup de chercheurs estiment que le codage d'un chromosome est la phase la plus importante pour un algorithme génétique vu son influence sur l'efficacité lors de la résolution du problème.

Dans la pratique, les structures peuvent être représentées sous différentes formes, des chaînes, des arbres, et des graphiques. Il existe également une variété de valeurs qui peuvent être attribués aux variables de décision, y compris des binaires, des symboles.

Souvent, les algorithmes génétiques utilisent des structures de chaîne contenant principalement des variables de décision binaires.

1	0	1	1	0	1	0	1

Figure 2.5. Codage binaire

D'après Holland, la représentation d'une solution sous forme d'une chaîne de bits est le codage le plus efficace. En effet, il permet d'avoir une solution plus longue ce qui offre plus de possibilités de croisements. Toutefois, ce type codage ne permet pas de représenter directement tous les problèmes.

Beaucoup de chercheurs préfèrent l'utilisation du codage symbolique. Il permet de représenter la solution d'une manière directe et plus simple.

B	A	F	C	E	M	P	O

Figure 2.6. Exemple de codage symbolique

La terminologie utilisée dans les AG est issue de la génétique. En effet, la structure de codage d'une solution est appelée « chromosome » ou « individu ». La variable de décision est appelée « gène » et sa valeur est appelée « allèle ».

3. La fonction Fitness

La spécification de la fonction d'évaluation ou de fitness est l'étape qui suit directement le codage. Le résultat fournit par la fonction fitness pour une solution donnée permet de décider si l'individu sera sélectionné ou non.

La valeur attribuée à chaque chromosome représente donc son aptitude ou sa capacité d'adaptation à l'environnement. L'algorithme génétique détermine les solutions qui ont les valeurs de fitness élevées parmi l'ensemble des solutions possibles.

Cette méthode permet de s'assurer que les individus performants seront conservés, alors que les individus peu adaptés seront progressivement éliminés de la population.

4. Les opérateurs

Une fois le codage et la fonction Fitness spécifiés, l'utilisateur doit choisir l'opérateur de sélection et les opérateurs génétiques pour générer des nouvelles solutions.

a. L'opérateur de sélection

L'opérateur de sélection assure la survie de la plus apte solution. Il existe plusieurs opérateurs mais ils partagent tous la même idée : La sélection des meilleurs parents de la population suivante en fonction de leur valeur de fitness. Il existe plusieurs méthodes de sélection. Nous citons ci-dessous les plus connues d'entre elles.

a.1. Sélection de la roulette (Roulette Wheel Selection)

C'est la plus classique et la plus utilisée. C'est une sélection proportionnelle basée sur le principe de la roulette. [Goldberg et Lingle, 1989]

La sélection de la roulette consiste à attribuer à chaque chromosome i, ayant une évaluation F_i, une probabilité de sélection P_i définie par l'équation suivante :

$$P_i = \frac{F_i}{\sum_i F_i} \qquad [2.1]$$

L'équation [2.1] illustre la probabilité de sélection si nous souhaitons maximiser la fonction d'évaluation. Par contre, si nous minimisons la fonction fitness P_i est définie par l'équation ci-dessous.

$$P_i = \frac{\frac{1}{F_i}}{\sum_i \frac{1}{F_i}} \qquad [2.2]$$

Lorsque la valeur de la fonction fitness des chromosomes varie énormément, la sélection de la roulette risque d'écarter des chromosomes ayant une probabilité de sélection faible.

Exemple :

Si la meilleure solution représente 90% de la roulette (P_i=90%) alors les autres individus auront très peu de chance d'être choisis et on arriverait à une stagnation de l'évolution

a.2. Sélection par rang de classement [Davis, 1991]

La sélection par rang de classement commence par trier les chromosomes selon un ordre croissant de la fonction d'évaluation et elle affecte le rang de chacun dans la population (le plus mauvais aura le rang 1).

Cette méthode est similaire à celle de la roulette, mais les portions sont en relation avec le rang plutôt qu'avec la fonction d'évaluation.

La sélection par rang améliore un peu la chance de sélection des moins bons individus mais sa convergence vers la bonne solution est lente.

a.3 Sélection par tournoi

Sur une population de N individus, p paires sont choisies au hasard. Parmi ces paires, le chromosome ayant le meilleur score d'adaptation est choisi.

Si la sélection était le seul opérateur, nous n'aurions jamais de nouvelles solutions. Afin d'explorer de nouvelles solutions, l'AG s'appuie sur deux opérateurs génétiques : le croisement et la mutation.

b. L'opérateur de croisement

Le croisement est un opérateur génétique permettant aux enfants d'hériter une partie du premier parent et l'autre du deuxième parent. Cette technique est généralement appliquée avec une probabilité élevée.

Dans la littérature, il existe plusieurs opérateurs de croisement qui dépendent essentiellement du type du codage et de la nature du problème à traiter [Mesghouni, 1999].

Nous présentons ci-dessous deux types d'opérateurs à savoir le croisement à un point et le croisement multipoints.

b.1. Le croisement à un point

Soient :

M : le nombre de gènes d'un chromosome

P1, P2 : la paire des parents sélectionnés

E1, E2 : les deux enfants générés suite à l'opération du croisement

Une position arbitraire k, compris entre 1 et M est choisie dans les deux parents P1 et P2. C'est le point de croisement. Comme indiqué dans la figure 2.4, Les gènes des parents P1 et P2 qui se trouvent avant le point de croisement sont copiés respectivement dans la structure des enfants (E1) et (E2). Les gènes manquants dans E1 et E2 sont remplis respectivement par ceux qui se trouvent dans P2 et P1, dans le même ordre d'apparition après le point de coupure

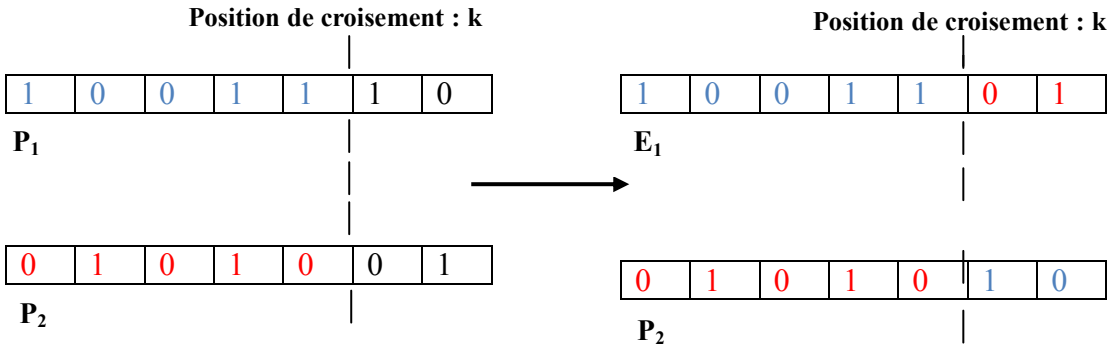


Figure 2.7. Le croisement à un point

b.2. Le croisement multipoints

Ce type de croisement applique les mêmes règles du précédent mais pour plusieurs points de croisement. La figure 2.8 décrit un croisement à 2 point C_1 et C_2

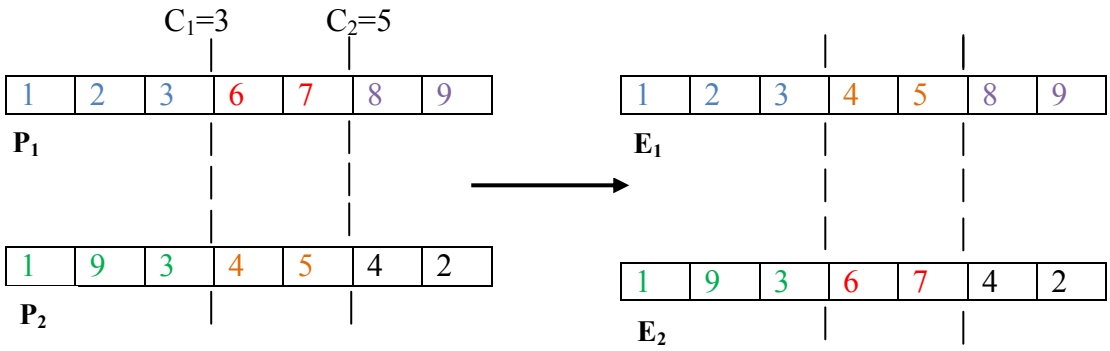


Figure 2.8. Le croisement à deux points

c. L'opérateur de mutation

L'opérateur de mutation est appliqué aux individus avec une probabilité faible. La mutation consiste à modifier la ou les valeurs d'un ou de plusieurs gènes d'un individu, sélectionnés aléatoirement. Cet opérateur favorise la diversification dans l'évolution des individus et évite les optimums locaux.

II. La recherche harmonique

1. Description

En 2000, Geem a développé une nouvelle métaheuristique nommée recherche harmonique. [Geem, 2000]

L'harmonie musicale est la combinaison de plusieurs sons simultanés ayant différentes fréquences. Comme le groupe des musiciens cherche à trouver l'harmonie musicale parfaite, déterminée par une norme esthétique en modulant l'état de chacun des instruments impliqués, le processus d'optimisation cherche la solution globale au problème, évaluée par la fonction objectif, en variant l'état ou les valeurs des variables de décisions.

Des exécutions musicales se répètent afin d'improviser l'harmonie. Les musiciens participants agissent sur leurs instruments pour lancer des tons dans la marge possible, créant ainsi l'ensemble des vecteurs d'harmonie.

Également, en optimisation, des itérations se poursuivent pour améliorer la solution, jugée selon sa fonction d'évaluation (Fitness). Dans chaque itération, des valeurs sont attribuées aux variables de décisions pour générer une configuration de solution.

La qualité de l'harmonie peut être améliorée d'un test à un autre et la valeur de la fonction objectif peut être aussi améliorée d'une itération à une autre.

Le tableau ci-dessous résume l'analogie entre l'improvisation de musique et le processus d'optimisation.

Tableau 2. 1. Comparaison entre le processus musical et le processus d'optimisation

Processus musicale	Processus d'optimisation
Harmonie musicale	Solution
Improvisation musicale	Itération
Instrument musical	Variable de décision
Ton	Valeur d'une variable de décision
Qualité de l'harmonie	Fonction objectif

Cette analogie est aussi illustrée dans la figure 2.9. Chaque instrument musical (saxophoniste, contrebassiste, guitariste) peut correspondre à une variable de décision (x_1, x_2 et x_3).

L'ensemble des notes de chaque instrument (saxophone = (Do, Ré, Mi); contrebasse = (Mi, Fa, Sol) et (guitare=(Sol, La, Si)) correspond à la liste de valeurs pouvant être attribuées aux variables de décision (x_1 = (100, 200, 300); x_2 = (300, 400, 500) et (x_3 = 500, 600, 700)).

Si le saxophoniste joue la note Do, la contrebasse Mi et le guitariste Sol, leurs notes en même temps font une nouvelle harmonie (Do, Mi, Sol). L'harmonie musicale est jugée ainsi

selon une norme esthétique. Si elle est meilleure que celles qui existent déjà, elle est mémorisée.

De même, la nouvelle solution (100mm, 300mm, 500mm) générée dans le processus d'optimisation est conservée si elle détermine une fonction fitness meilleure que les autres solutions existantes.

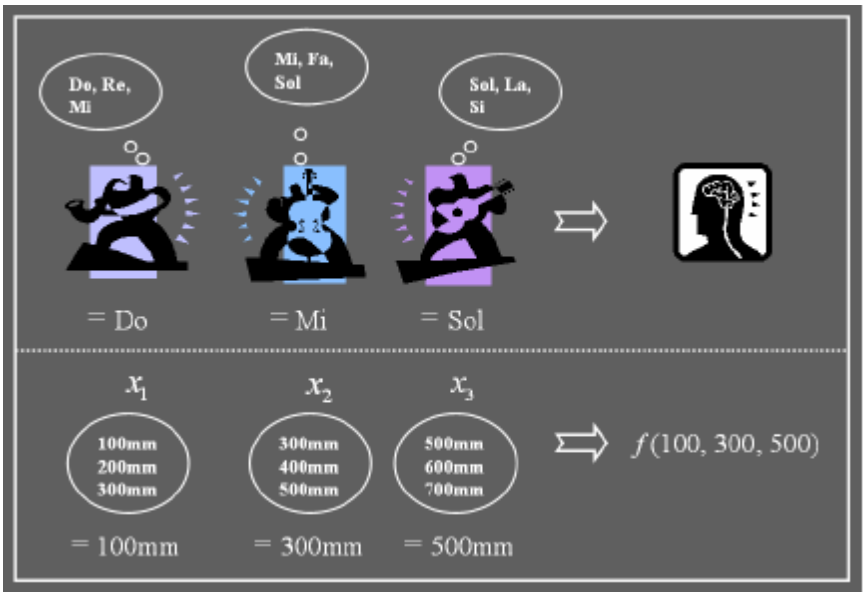


Figure 2.9. L'analogie entre les improvisations musicales et l'optimisation [Lee et Geem, 2005]

2. La démarche d'un algorithme de recherche harmonique

L'algorithme RH comprend cinq étapes:

- Initialisation des paramètres,
- Initialisation de la mémoire harmonique,
- Improvisation d'une nouvelle harmonie,
- Mise à jour de la mémoire harmonique
- Vérification du critère d'arrêt.

a. Initiation des paramètres :

Dans la première étape, le problème d'optimisation est spécifié comme suit :

Minimiser (ou Maximiser) f (**x**), avec $x_i \in X_i$, i = 1,2,..., N

Où :

- f(x) est la fonction objectif
- x est le vecteur de solutions composé des variables de décision x_i
- X_i est l'ensemble des valeurs possibles pour la variable x_i
- N est le nombre de variables de décision (nombre d'instruments musicaux)

- K est le nombre valeurs possibles pour chaque variable de décision discrète (nombre de tons pour chaque instrument).

Les paramètres de l'algorithme sont aussi spécifiés durant cette étape :

- La taille de la mémoire harmonique, c'est le nombre de solutions dans la mémoire (HMS)
- Le taux de considération de la mémoire harmonique (HMCR)
- Le taux de réglage de tons (PAR)
- Le critère d'arrêt (le nombre des improvisations).

HMCR et PAR sont deux paramètres utilisés dans la troisième étape pour créer le vecteur de solution.

b. L'initialisation de la mémoire harmonique:

Au cours de cette étape, la mémoire harmonique est générée. A chaque variable de décision est attribuée d'une façon aléatoire une valeur de sa liste de choix. A chaque solution, il correspond une valeur de la fonction fitness. [Mahdavi et col., 2007]

$$HM=\begin{bmatrix} x_1^1 & x_2^1 & \dots & x_{N-1}^1 & x_N^1 & \Rightarrow f(x^1) \\ x_1^2 & x_2^2 & \dots & x_{N-1}^2 & x_N^2 & \Rightarrow f(x^2) \\ . & \dots & \dots & \dots & \dots & \\ . & \dots & \dots & \dots & \dots & \Rightarrow \dots \\ . & \dots & \dots & \dots & \dots & \\ x_1^{HMS-1} & x_2^{HMS-1} & \dots & x_{N-1}^{HMS-1} & x_N^{HMS-1} & \Rightarrow f(x^{HMS-1}) \\ x_1^{HMS} & x_2^{HMS} & \dots & x_{N-1}^{HMS} & x_N^{HMS} & \Rightarrow f(x^{HMS}) \end{bmatrix} \quad [2.3]$$

c. L'improvisation d'une nouvelle harmonie

Un nouveau vecteur harmonique $x = (x_1, x_2, \dots, x_N)$ est produit en utilisant les trois règles suivantes:

- La sélection arbitraire
- La considération de la mémoire harmonique
- Le réglage du ton

c.1. La sélection arbitraire:

HMCR (Harmony Memory Consideration Rate) est un taux de considération de la mémoire harmonique qui varie entre 0 et 1. Il est généralement entre 70% et 99%.

Les valeurs des variables de décision x_i' sont choisies, avec une probabilité (w.p) HMCR, à partir des valeurs stockées dans HM et elles sont choisies aléatoirement à partir de la gamme possible des valeurs pour une probabilité (1-HMCR).

$$x_i' \leftarrow \begin{cases} x_i' \in \left\{ x_i^1, x_i^2, \ldots, x_i^{HMS} \right\} & \text{w.p} \quad HMCR \\ x_i' \in X_i & \text{w.p} \quad (1 - HMCR) \end{cases} \qquad [2.4]$$

La figure ci-dessous présente le choix d'une valeur pour la composante x_1' de la variable de décision x'.

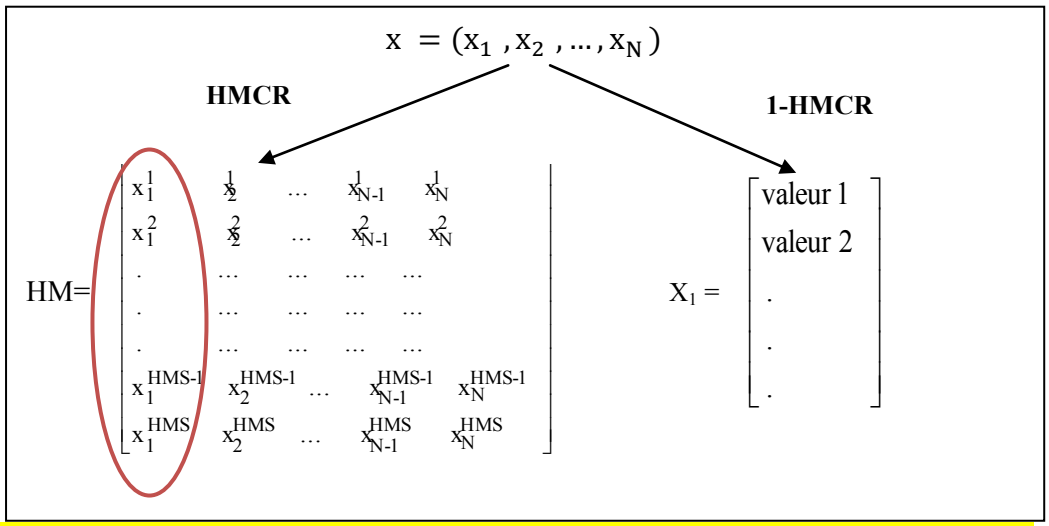


Figure 2.10. Exemple de choix d'une valeur pour une composante de la variable de décision

Lorsque la mémoire harmonique est utilisée dans le choix d'une valeur pour une composante de x_i , un second paramètre d'ajustement est à considérer, le PAR (Pitch Adjusting Rate). Il varie généralement entre 10% et 50%

c.2. La considération de la mémoire harmonique

Chaque composante dans le nouveau vecteur harmonique x', déterminé à partir de la mémoire harmonique (avec une probabilité HMCR), est examinée pour déterminer si elle doit être ajustée. Une légère variation additive est ainsi attribuée pour ses valeurs.[Mahdavi et col., 2007]

$$x_i' \leftarrow \begin{cases} x_i' \pm \text{rand}() * bw & \text{w.p} \quad HMCR \times PAR \\ x_i' & \text{w.p} \quad HMCR \times (1 - PAR) \end{cases} \qquad [2.5]$$

Le mécanisme de création est purement aléatoire pour une probabilité HMCR× (1-PAR), x_i garde sa valeur.

Par contre, s'il s'agit d'une probabilité HMCR × PAR, x_i est ajustée et sa valeur est égale à $x_i' \pm \text{rand}() * bw$

avec :

bw : une distance arbitraire (bandwidth)

rand () : un nombre aléatoire entre 0 et 1

d. La violation de la considération harmonique :

La nouvelle solution doit être vérifiée pour voir si elle respecte les contraintes de la problématique

e. La mise à jour de la mémoire harmonique :

La nouvelle solution est ajoutée à la mémoire harmonique si elle est meilleure que la plus mauvaise solution harmonique dans la HM en terme de fitness et cette dernière est supprimée de la HM.

f. la vérification du critère d'arrêt :

Si le critère d'arrêt est atteint, le calcul est terminé. Sinon, les étapes (c) et (d) sont répétés. La figure ci-dessous résume la démarche suivie par l'algorithme de la recherche harmonique

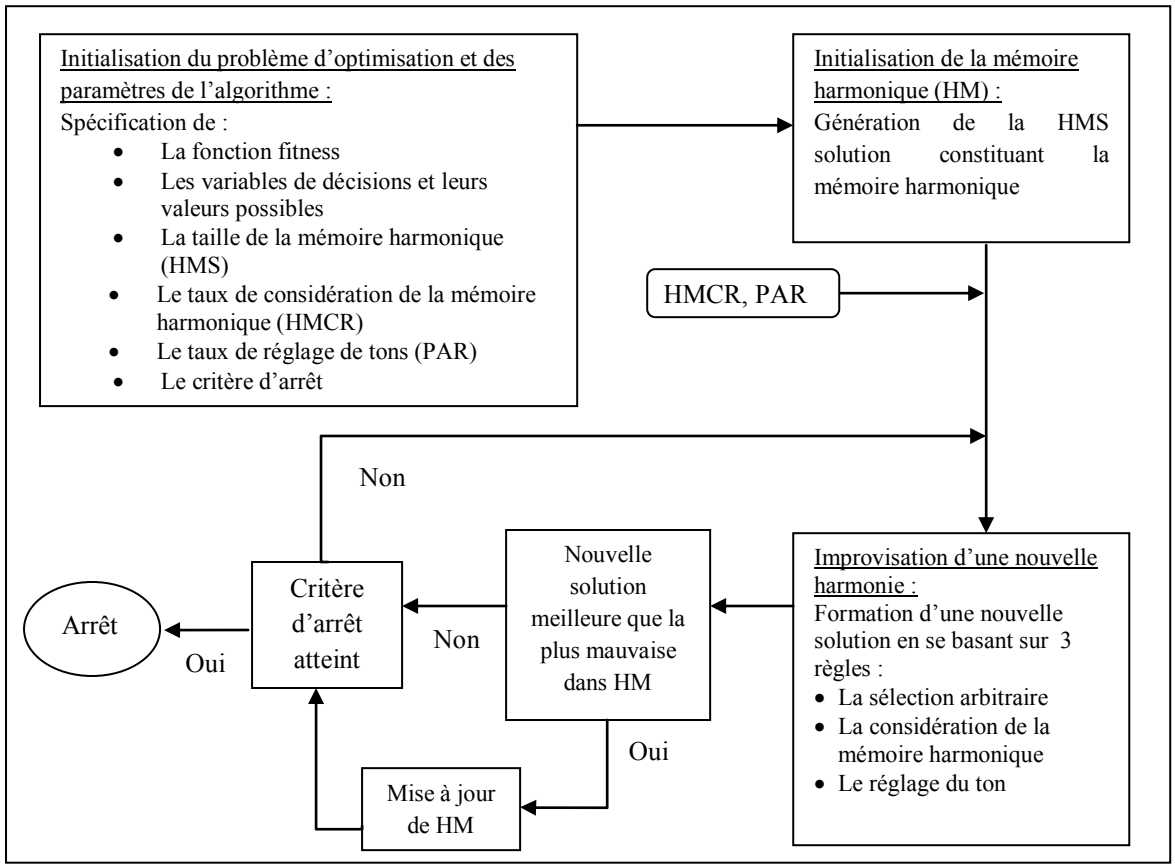


Figure 2.11. La procédure de la recherche harmonique [Lee et Geem, 2005]

3. Les applications de la recherche harmonique

Depuis son apparition en 2000, la recherche harmonique a été utilisée dans beaucoup de publications. Elle a été appliquée à la conception d'un réseau de tayaux [Geem et col., 2000, Geem et col., 2002], à la conception des réseaux de distribution d'eau [Geem, 2000, Geem,

2006, Geem, 2007, Geem, 2008], aux tournées de véhicule [Geem et col., 2005a], au problème de course d'orientation généralisé [Geem et col., 2005b], à la conception des systèmes de grillage [Erdal et Saka, 2008], à l'emploi du temps de cours universitaire [Al-Betar et col., 2008], à l'optimisation des coûts d'amarrage [Ryu et d'autres., 2007], etc

Plusieurs extensions de l'algorithme de base ont été aussi proposées, le paramétrage d'un algorithme dynamique [Mahdavi et col., 2007], la prévention de la meilleure solution [Omran et Mahdavi, 2008], la variante dynamique [Geem et Hwangbo, 2006] et une hybridation avec programmation quadratique séquentielle [Fesanghary et col., 2008].

La recherche harmonique a été appliquée avec succès dans presque tous les travaux déjà cités. Le nombre croissant d'applications suggère que la recherche harmonique HS soit une heuristique prometteuse pour de nouvelles recherches.

L'algorithme (RH) ne nécessite pas de valeurs initiales pour les variables de décision. En outre, au lieu d'un gradient de recherche, le RH utilise un algorithme stochastique de recherche aléatoire qui est basé sur le taux de considération de la mémoire harmonique (HMCR) et le taux de réglage (PAR) de sorte que les dérivées de l'information sont inutiles. En comparant le RH aux métaheuristiques connues antérieurement, nous constatons qu'il impose moins d'exigences mathématiques et peut facilement être adopté pour les différents types de problèmes d'optimisation. [Lee et Geem, 2005]

4. Exemple d'application de la recherche harmonique :

L'algorithme de la recherche harmonique est appliqué pour déterminer la solution optimale $X= [x_1 \ x_2]^T$ qui minimise la fonction suivante :

$$\text{Minimiser } F(x)= 5 x_1^2 - 9 x_1x_2 + 5x_2^2$$

$$\text{Sujet à } G(x) =25-16 x_1x_2 \leq 0$$

Les valeurs de x1 et x2 sont limitées à l'ensemble {0.5, 1.0, 1.5, 2.0, 2.5, 3.0,…,10.0}.
La fonction objectif est f (x), qui est l'une des fonctions de test standard pour les problèmes d'optimisation [Jenkins, 1992]. L'optimum discret du problème est X= [1.5 1.5] T, ce qui a été vérifiée à partir des conditions de Kuhn-Tucker.

L'algorithme de la recherche harmonique a été appliqué pour résoudre cette fonction. La taille de la mémoire harmonique HMS ou le nombre total de vecteur solution est fixée à 10. Le Taux de considération de la mémoire harmonique (HMCR) est sélectionné comme 0,9 tandis que le taux d'ajustement est considéré comme 0,2. La matrice harmonique initiale ou la population initiale est présenté dans le tableau suivant.

Tableau 2.2. Mémoire harmonique initiale

N° Vecteur	x_1	x_2	F(x)
1	3.0	4.5	24.75
2	2	4	28
3	7.5	6	56.25
4	4.5	7	62.75
5	5.5	8	75.25
6	8.5	5.5	91.75
7	10	9	95
8	10	10	100
9	5.5	1	106.7
10	1	9	329

Comme le montre le tableau 2.2, la mémoire harmonique est générée aléatoirement. Les vecteurs solutions sont triés selon les valeurs de la fonction objectif

Les $11^{ème}$, $12^{ème}$ et $13^{ème}$ essais pour l'improvisation d'une nouvelle harmonie n'ont pas trouvé une solution meilleure que ceux illustrés dans tableau 2.2. Cependant, la $14^{ème}$ recherche a donné une meilleure solution comme le montre le tableau 2.3.

Le vecteur de la nouvelle harmonie est $X= [6, 9]^T$. Il a été improvisé en utilisant une probabilité (HMCR) de 72,0%, un taux d'ajustement (PAR) 0,18%. Comme la valeur de fonction objectif de la nouvelle harmonie $[6, 9]^T$ est 99.00, la nouvelle harmonie est incluse dans la mémoire harmonique (HM) et la plus mauvaise solution $[1, 9]^T$ est exclue de la HM, comme le montre le tableau 2.3

Tableau 2.3. Mémoire harmonique après 14 essais d'improvisation d'une nouvelle solution

N° vecteur	x_1	x_2	F(x)
1	3	4.5	24.75
2	2	4	28
3	7.5	6	56.25
4	4.5	7	62.75
5	5.5	8	75.25
6	8.5	5.5	91.75
7	10	9	95
8	6	9	99

9	10	10	100
10	5.5	1	106.75

L'algorithme de recherche d'harmonie continue la recherche d'une meilleure combinaison, Le tableau 2.4 présente les harmonies générées après 50 recherches.

Tableau 2.4. Mémoire harmonique après 50 essais d'improvisation d'une nouvelle solution

N° vecteur	x_1	x_2	F(x)
1	1.5	1.5	2.25
2	2	1.5	4.25
3	2.5	2.5	6.25
4	3	2.5	8.75
5	3	3	9
6	2	3	11
7	3	3.5	11.75
8	3.5	3.5	12.25
9	2.5	3.5	13.75
10	3.5	4	15.25

Plus le nombre de générations est grand plus on se rapproche du vecteur optimum, $x^* = [1.5 \ \ 1.5]^T$. L'harmonie optimale est déterminée après 50 recherches. Les résultats finaux obtenus après 50 recherches sont présentés dans le tableau 2.4.

Conclusion

Dans ce chapitre, deux approches de résolution ont été détaillée, l'algorithme génétique et la recherche harmonique. Ces deux techniques vont être adaptées pour la résolution des différentes variantes du problème de stockage de conteneurs décrites dans les trois prochains chapitres.

Chapitre 3 : Résolution du PSC statique pour un seul et plusieurs types de conteneurs

Avoir entendu sans retenir ne fait pas de la science.
[Dante Alighieri]

Introduction

La gestion d'un terminal à conteneurs comporte différentes tâches allant de l'affectation des navires aux postes d'amarrage et de leur déchargement jusqu'au chargement des conteneurs pour être transportés par d'autres navires ou par voies ferroviaires et/ou routières et inversement. Entre leur arrivée et leur départ du port, les conteneurs sont attribués temporairement aux emplacements disponibles dans les différents blocks de stockage. Cette opération est très importante vu qu'elle a une influence directe sur le processus de récupération des conteneurs destinés à l'import et à l'export lors de leurs départs. En effet, une meilleure répartition des conteneurs dans les blocks de stockage optimise le processus de déchargement et minimisera les opérations de remaniement. Ceci diminue la durée de cette tâche (throughput, Truck turnaround time) et aussi les retards des véhicules de transport.

Dans ce chapitre, nous allons se concentrer sur le processus de chargement des conteneurs dans les blocks de stockage où un seul type de conteneur (le type général) va être considéré dans la première partie et plusieurs types dans la seconde.

Pour résoudre ces problèmes, deux méthodes de résolution sont utilisées, un algorithme génétique et un algorithme de recherche harmonique.

Ce chapitre comporte six sections. La première est consacrée à la présentation du PSC où un seul type de conteneurs est considéré, sa définition et sa formulation mathématique sont détaillées. La section deux et trois sont dédiés à l'adaptation de l'algorithme génétique (AG) et de l'algorithme de recherche harmonique (RH) au problème. Le reste du chapitre est consacré à l'étude du PSC à plusieurs types. Nous présentons la formulation mathématique ainsi que l'application des AG et de la RH au problème.

I. Présentation du PSC à un seul type de conteneurs

1. Définition du problème

Les conteneurs destinés à être transportés par les navires ou par voies ferroviaires ou routières sont stockés dans les différents blocks de stockage. Le problème de stockage de conteneurs (PSC) décrit ce processus. L'objectif principal de ce problème est de trouver un

arrangement des conteneurs arrivant au port, destinés à l'import ou à l'export, parmi les emplacements disponibles dans les différents blocks de stockage, qui minimise les opérations de remaniement lors de leurs départs des zones de stockage.

En effet, l'organisation doit tenir compte des dates d'arrivée et surtout des dates de départ de conteneurs de l'espace de stockage. En plus, il doit prendre en considération l'état initial des blocks de stockage qui contiennent déjà des conteneurs.

2. Formulation mathématique du problème

Dans cette sous-section, la formulation du problème décrit ci-dessus est détaillée. D'abord, quelques suppositions concernant le problème sont données. Ensuite, les différents paramètres utilisés dans la formulation sont définis. Enfin, la formulation mathématique du problème est décrite.

a. Suppositions :

Pour résoudre ce problème, un nombre de suppositions doit être pris en considération :

- Tous les conteneurs sont identiques ; même type, même forme et même poids.
- Un conteneur ne peut pas être déchargé directement. En effet, tous les conteneurs se trouvant directement en dessus doivent être déchargés au préalable.

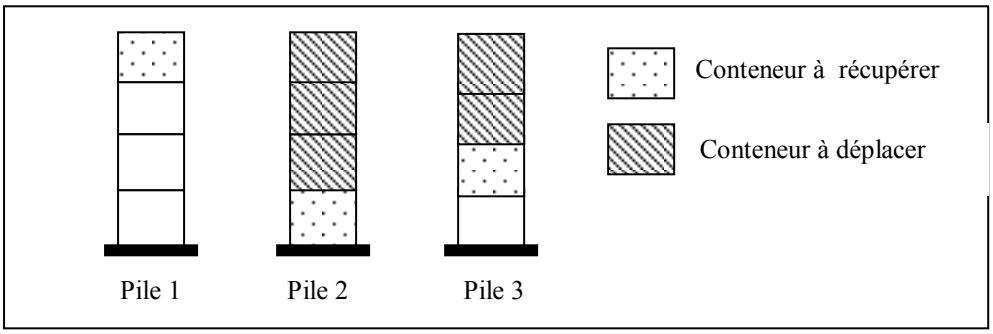


Figure 3.1. Les mouvements de remaniements

- Les blocs de stockage sont initialement vides.
- Tous les conteneurs vont être stockés dans des blocks à trois dimensions. Ce système peut être décrit par la figure suivante :

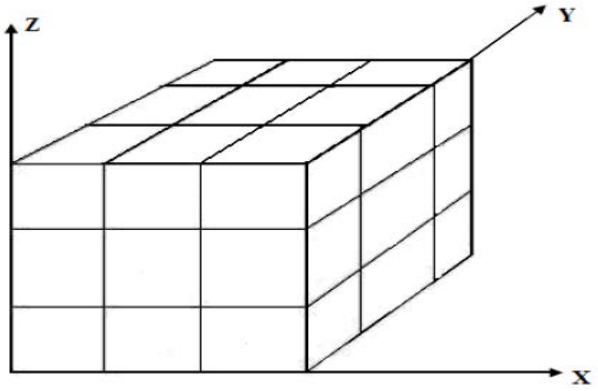


Figure 3.2. Système de coordonnées cartésiennes

b. Définition des paramètres :

Les différents paramètres du problème peuvent être définis comme suit :

- i : indice du conteneur
- N_{Bloc}: nombre de blocks disponibles
- b : indice d'un bloc ; $b = 1, \ldots, N_{Bloc}$
- N_c : nombre de conteneurs stockés dans les blocs.
- $Nc_{Etage}(j,b)$: nombre de conteneurs à l'étage j du block b.
- d_i : date de départ du conteneur i
- n_1 : le nombre maximum de conteneurs sur l'axe des X
- n_2 : le nombre maximum de conteneurs sur l'axe des Y
- n_3 : le nombre maximum de conteneurs sur l'axe des Z (nombre d'étages)

c. Formulation mathématique :

Dans ce paragraphe, la formulation mathématique du PSC est présentée. L'objectif principal de ce problème est de minimiser le nombre de mouvements de remaniement des conteneurs se trouvant dans la zone de stockage, lors de leurs départs. Le problème décrit ci-dessus et tenant compte des suppositions sus-indiquées, peut être formulé comme suit :

Sous contraintes:

$$\mathrm{Min} \sum_{i=1}^{Nc} \sum_{b=1}^{N_{Bloc}} m_i C_i\ (x,y,z,b); \quad \forall x = 1..n_1, \forall y = 1..n_2, \forall z = 1..n_3 \qquad [3.1]$$

$$Nc_{Etage}(j,b) \geq Nc_{Etage}((j+1),b), \text{ pour } j = 1...n_3 - 1,\ b = 1...N_{Bloc} \qquad [3.2]$$

$$C_i(x,y,z,b) - C_i(x,y,z+1,b) \geq 0, \forall i \in [1..Nc] \qquad [3.3]$$

Avec:

- m_i : Le nombre minimum de conteneurs manipulés pour retirer le conteneur i

- C_i *(x, y, z, b),* la variable de décision, définie par :

$$C_i(x, y, z, b) = \begin{cases} 1, \text{ Si le conteneur i est à la position } (x, y, z) \text{ du block b} \\ 0, \text{ Sinon} \end{cases} \quad [3.4]$$

Les contraintes données par les équations 3.2 et 3.3 veillent à ce qu'un étage contienne au moins le même nombre de conteneurs que celui directement eu dessus. Elles illustrent également le fait qu'un conteneur ne peut avoir que deux positions, soit sur un autre soit sur terre.

Pour résoudre ce problème, plusieurs travaux ont été proposés (Preston et Kozan (2001), Chen et ses collègues (2004), Kim et Park (2003), Zhang et ses collègues (2003), etc.).

Dans cette thèse, deux approches sont proposées, un algorithme génétique et une recherche harmonique. L'adaptation de chacune de ces deux techniques au PSC statique d'un seul type est détaillée dans les sections 2 et 3.

II. Adaptation de l'algorithme génétique au PSC à un seul type de conteneurs

1. Introduction :

L'algorithme génétique est une méthode évolutionnaire composée de six principales étapes. Lors de la première, le codage d'une solution au problème est donné. Ensuite, une population initiale est construite, généralement, de façon aléatoire. Puis, deux solutions, appelées parents, sont sélectionnées pour subir un éventuel croisement pour donner une ou plusieurs nouvelles solutions appelées enfants. Enfin, un opérateur de mutation peut être appliqué à ces enfants. Ce processus est itéré jusqu'à atteindre un critère d'arrêt qui peut être ou bien un nombre maximal de répétitions ou un nombre d'itérations sans amélioration de la meilleur solution.

Pour être adapté au PSC, l'AG développé dans cette thèse peut être décrit comme suit : initialement *N* solutions sont générées de façon aléatoire. Par la suite, deux parents sont sélectionnés en utilisant la méthode de la roulette. Un croisement à un point est appliqué aux deux parents, avec une probabilité *Pcrois* à définir, pour générer deux enfants. Puis, avec une probabilité *Pmut*, les deux enfants vont subir une mutation. Ces trois dernières étapes (sélection, croisement et mutation) sont exécutées pour créer *N* nouvelles solutions pour former une population intermédiaire, P_{inter}, de taille *2N*. Enfin, les solutions de P_{inter} sont triées dans l'ordre croissant des valeurs de leur fonction objectif et les *N* meilleurs solutions formeront la population initiale pour l'itération suivante. Ce processus est itéré jusqu'à atteindre le critère d'arrêt. Ces différentes étapes sont détaillées dans la sous-section suivante.

2. Description de l'AG pour le PSC à un seul type

a. Codage d'une solution

Le codage permet de représenter les solutions sous forme de chromosomes. Un chromosome décrit la disposition des conteneurs dans les différents blocks de stockage. La figure représente un exemple d'arrangement des conteneurs dans un bloc.

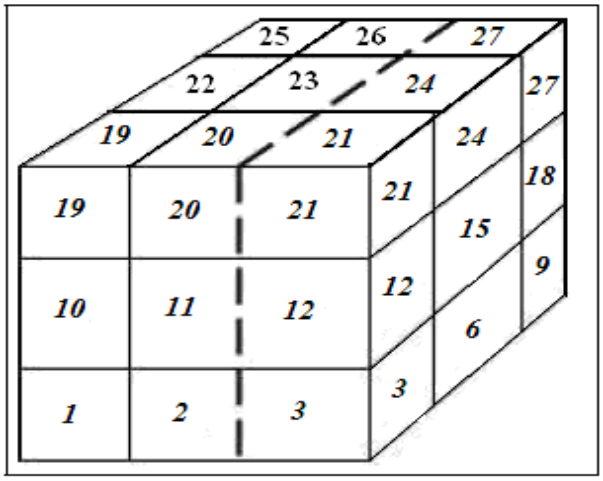


Figure 3.3. Répartition des conteneurs dans un bloc (codage du chromosome)

b. Génération de la population initiale

L'algorithme utilisé pour la création de la population initiale génère des solutions valides (satisfaisant les contraintes 3.1 et 3.2). Il cherche dans un bloc sélectionné de façon aléatoire, l'emplacement disponible pour affecter un conteneur choisi d'une façon aléatoire à cette place.

Algorithme génération population initiale
Début
// Nc : Le nombre de conteneurs
//Nbrpop : La taille de la population
pop=0
Faire
  Pour i = 1 à Nc
    - Choix arbitraire d'un conteneur c
    - Sélection aléatoire d'un bloc ayant des places vides
    - Sélection arbitraire d'un emplacement vide du bloc et respectant les contraintes
    - Affecter le conteneur c à cet emplacement

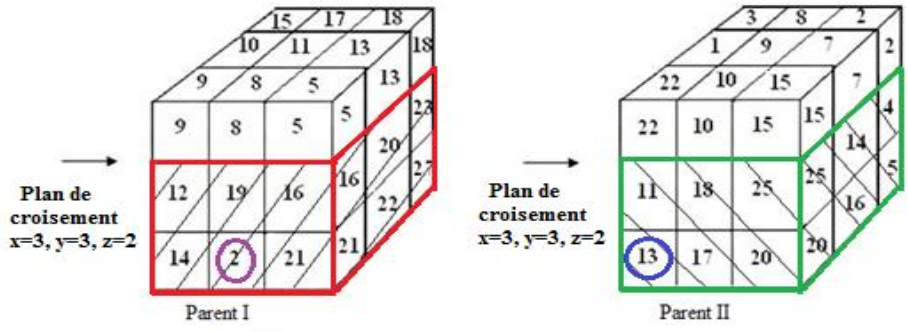
  Fin Pour
Si la nouvelle solution n'existe pas
//Ajouter la nouvelle solution à la population
    pop++
Tant que (pop < Nbrpop)
Fin

Figure 3.4. Algorithme création de la population initiale pour les conteneurs identiques

c. Croisement

L'opérateur de croisement est appliqué à deux individus I_1 et I_2 sélectionnés par la méthode la roulette, pour générer deux enfants E_1 et E_2. L'opérateur adopté est le croisement à un

point. Il consiste à sélectionner selon les axes *X*, *Y* et *Z,* trois point d'intersection *p-crois-x*, *p-crois-y* et *p-crois-z* pour former le plan de croisement pour chaque block de stockage.



Parent 1

Etage 1(z=1)		
1	**3**	27
25	4	**22**
14	**2**	21

Etage 2 (z=2)		
24	6	23
7	26	20
12	19	16

Etage 3 (z=3)		
15	17	18
10	11	13
9	**8**	5

Parent 2

Etage 1(z=1)		
27	21	**5**
12	24	16
13	**17**	20

Etage 2 (z=2)		
23	19	4
6	26	14
11	**18**	25

Etage 3 (z=3)		
3	**8**	2
1	**9**	7
22	**10**	**15**

Enfant 1

Etage 1(z=1)		
1	3	27
25	4	22
14	2	21

Etage 2 (z=2)		
24	6	23
7	26	20
12	19	16

Etage 3 (z=3)		
15	**9**	**8**
11	**18**	**10**
13	**17**	**5**

Enfant 2

Etage 1(z=1)		
27	21	5
12	24	16
13	17	20

Etage 2 (z=2)		
23	19	4
6	26	14
11	18	25

Etage 3 (z=3)		
8	**10**	**15**
3	**7**	**9**
2	**22**	**1**

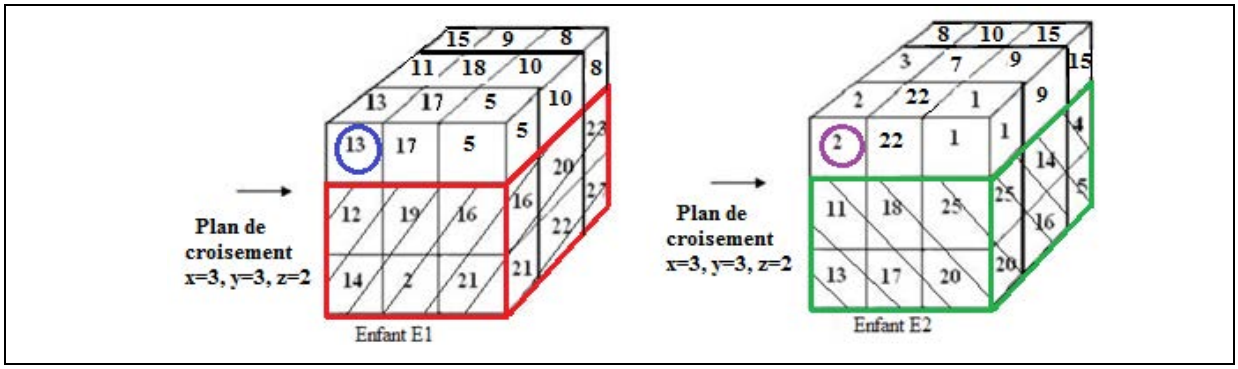


Figure 3.5. Opération de croisement

Par la suite, l'enfant E_1 recevra les gènes (conteneurs) de l'individu I_1 situés dans le plan de croisement et les conteneurs restants seront affectés dans le reste des emplacements vides selon l'ordre de leur apparition dans l'individu I_2. De même, l'enfant E_2 recevra les gènes de l'individu I_2 situés dans le plan de croisement et les autres conteneurs sont affectés dans le reste des emplacements vides selon l'ordre de leur apparition dans l'individu I_1.

La figure suivante décrit l'opérateur de croisement appliqué à deux individus I_1 et I_2 pour générer deux enfants E_1 et E_2. Dans cet exemple, où un seul bloc est considéré, le plan de croisement est défini par *p-crois-x* =3, *p-crois-y*=3 et *p-crois-z* =2.

Il est à noter que l'opérateur de croisement est exécuté avec une probabilité à définir. Cette probabilité est généralement entre 70% et 95%. Les deux enfants qui peuvent être générés après croisement sont ajoutés à la population s'ils n'y existent pas déjà

Algorithme de croisement pour un seul type
Début
- Choix arbitraire d'une probabilité : Prbc

Si Prbc < 80%
- Sélection de 2 parents (I1, I2) en utilisant la méthode de la roulette
- Choisir au hasard 3 coordonnées définissant un plan de croisement (*p-crois-x*, *p-crois-y* et *p-crois-z*) respectivement dans [0, .., n_1-1], [0,…,n_2-1]et [0,…,n_3-1]
- Copier les conteneurs occupants dans I1, le plan [0, p-crois-x],[0, p-crois-y] et [0, p-crois-z] dans l'enfant E1.
- Déterminer la liste des conteneurs qui ne sont pas encore stockés dans E1
  // La taille de la liste est Nc_{rest}, le conteneur qui reste noté r ∈ [1, Nc_{rest}]

  Répéter
  - ✓ Trouver l'emplacement du conteneur r dans I2
  - ✓ Si l'emplacement est convenable pour E1
    Stocker le conteneur r
    Sinon
    Chercher le plus proche emplacement adéquat et stocker le conteneur r
  - ✓ Mise à jour de la liste des conteneurs restant

  Tant que (r ≤ Nc_{rest})

Fin

Figure 3.6. Algorithme de croisement pour des conteneurs identiques

d. Mutation

Les deux enfants crées après croisement subissent une opération de mutation avec une probabilité assez faible (en générale nettement inférieure à 30%). Cet opérateur tend à appliquer une modification minuscule sur les informations génétiques d'une solution. Ses principaux objectifs sont de favoriser la diversification des solutions par l'exploration de plusieurs espaces de recherche d'une part, et de sortir des optimums locaux d'autre part.

L'opération de mutation adoptée dans cette approche consiste à permuter deux conteneurs dans chaque block de stockage.

Algorithme de Mutation pour un seul type
Début
- Choix arbitraire d'une probabilité : Prbm
- Si (Prbm < 20%)
  Pour c=0 to Nbloc
  - ✓ Sélection arbitraire de 2 conteneurs du bloc c
  - ✓ Permuter les 2 conteneurs

  Fin Pour

Fin

Figure 3.7. Algorithme de mutation pour un seul type de conteneurs

La figure 3.8 décrit la disposition des conteneurs dans un block de stockage avant et après mutation. Dans cette figure, les conteneurs de coordonnées (1, 1,3) et (2, 1,1) sont permutés.

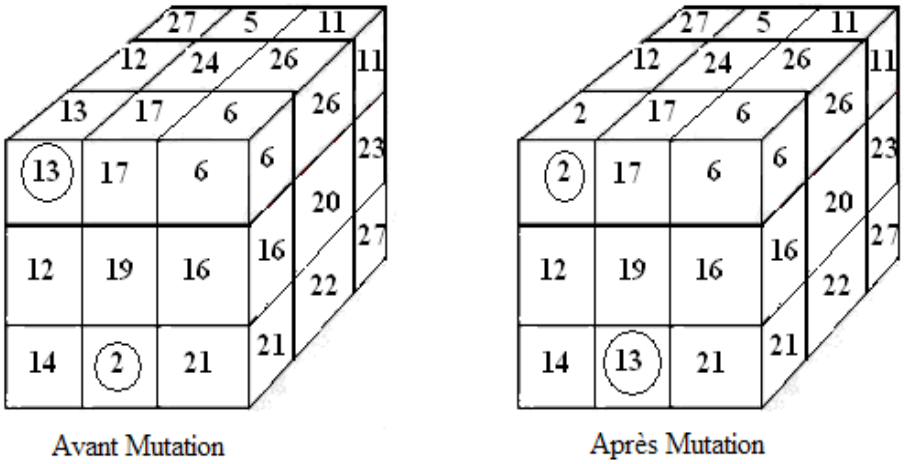


Figure 3.8. Opération de mutation

e. Critère d'arrêt

Ce critère définit la condition d'arrêt du processus d'optimisation et d'acceptation de la meilleure solution générée. Pour cette adaptation, le processus de sélection, croisement et mutation s'arrête après un nombre de générations à définir.

III. Adaptation de l'algorithme de recherche harmonique au PSC à un seul type

1. Introduction

La recherche harmonique (RH) est une méta-heuristique qui a été développée pour imiter le comportement d'un musiciens en quête de la meilleure harmonie, en se basant l'improvisation. Cette approche comprend cinq étapes. D'abord, les paramètres nécessaires pour la résolution du problème sont initialisés (définition des variables de décision et des objectifs et spécification de paramètre de la RH (HMS, HMCR, PAR, etc.)). Puis, la mémoire harmonique est initialisée pour former la population initiale. Ensuite, une nouvelle harmonie (solution) est créée. Elle prend par la suite la place de la plus mauvaise solution de la population, si elle est meilleure qu'elle. Ces deux dernières étapes se répètent jusqu'à atteindre le critère d'arrêt. L'adaptation de cet algorithme pour le PSC est détaillée dans la prochaine sous-section.

2. Description de l'algorithme de RH pour le PSC à un seul type

a. Initialisation des paramètres :

Dans cette adaptation, le vecteur X des variables de décision est composé par l'ensemble des conteneurs i à stocker.

$$X = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ . \\ . \\ . \\ Nc \end{bmatrix} \qquad [3.5]$$

L'ensemble des valeurs possibles pour la variable de décision i est noté X_i .C'est l'ensemble de tous les emplacements possibles (vides) pour stocker le conteneur i

Exemple : Si le bloc 0 est vide

X_i= {cont[0][0][0][0], .. , cont[n1-1][n2-1][n3-1][0]}

avec cont = {cont[x][y][z][b]; $1 \leq x \leq n1$, $1 \leq y \leq n2$, $1 \leq z \leq n3$, $1 \leq b \leq B$ } désigne un emplacement définie par les coordonnées (*x, y, z*) dans un block *b*.

Une mise à jour de la liste des emplacements X_i est effectuée après l'affectation d'un conteneur.

b. Initialisation de la population initiale

La population initiale qui forme la mémoire harmonique (HM) est générée en utilisant la même que celle utilisée pour créer la population initiale dans l'AG.

c. Création d'une nouvelle solution

La RH peut être considéré comme un algorithme constructif, c'est-à-dire qui construit la solution itération par itération. Dans chaque itération, un conteneur i est choisi aléatoirement de la liste des conteneurs à stocker. Si la probabilité est inférieure à HMCR, i est affecté à un même emplacement *cont[x][y][z][b]* qu'occupe ce même conteneur dans une ancienne solution de la mémoire harmonique. Avec une probabilité supérieure à HMCR, le conteneur i est affecté à une position aléatoire de sa liste de choix (X_i). Le choix arbitraire des positions doit aussi considérer les contraintes [3.1] et [3.2].

Chaque conteneur occupant une place sélectionné à partir de la mémoire harmonique subit une opération d'ajustement avec une probabilité PAR. Cette opération favorise l'exploration de nouvelles solutions et vise à éviter les optimums locaux.

Ce processus est itéré jusqu'à affectation de la totalité des conteneurs.

Algorithme Création d'une nouvelle solution

Début

// Le nombre de conteneurs à stocker est Nc

n= 0 ;

Répéter

- Choix arbitraire d'un conteneur : i
- Choix arbitraire d'une probabilité : P_{HMCR}
- Si $P_{HMCR} < 90\%$
  - ✓ Choix arbitraire d'une solution S de la population initiale
  - ✓ Si l'emplacement du conteneur i dans S est convenable pour la nouvelle solution
    - ▪ Stocker le conteneur i dans cette place
    - ▪ Choix arbitraire d'une probabilité : P_{PAR}
    - ▪ Si $P_{PAR} < 10\%$
      - Appliquer la procédure d'ajustement
      - Stocker le conteneur c dans la position ajustée
      - n = n+1
- Sinon
  - ✓ Sélectionner aléatoirement un emplacement convenable pour le conteneur i à partir de sa liste des positions
  - ✓ Stocker le conteneur i
  - ✓ n = n+1

Tant que (n ≤ Nc)

Fin

Figure 3.9. Algorithme de création d'une nouvelle solution

d. Mise à jour de la mémoire harmonique

Après avoir créé une nouvelle solution, celle-ci est ajoutée à la HM si elle est meilleure que la plus mauvaise solution déjà trouvée.

e. Vérification du critère d'arrêt

Le processus de création d'une nouvelle harmonie et l'opération de mise à jour de la HM sont répétés pendant un nombre de générations à définir par l'utilisateur.

IV. Présentation du PSC pour plusieurs types de conteneurs

1. Introduction

Le PSC, a été étudié dans les sections précédentes, comme dans la plupart de la littérature qui le traite, considérant qu'un seul type de conteneur (général), malgré qu'il en existe plusieurs. Les travaux traitant le PSC à plusieurs types de conteneurs sont rares, vu que chaque type possède un ensemble de contraintes concernant son stockage. Ceci complique la formulation du problème et rend sa résolution beaucoup plus difficile.

Les conteneurs peuvent être classés selon la forme ou le type. D'un point de vue forme, on distingue le conteneur à toit ouvert, plate-forme, citerne, etc. D'un point de vue type, les conteneurs diffèrent selon les marchandises qu'ils peuvent stocker. On trouve les conteneurs généraux (conteneur à usage général ou à usage spécifique) qui peuvent être placés dans n'importe quel bloc de stockage et les conteneurs pour marchandises spécifiques (conteneur ventilé ou à température contrôlée) qui doivent être stockés dans des blocs alimentés par l'électricité.

De ce fait, deux types de blocs de stockage sont aussi considérés, les blocs standards et les blocs réfrigérés (alimentés par l'électricité).

Au niveau de cette thèse, six familles de conteneurs sont considérées. Chacune d'entres elle, possède un ensemble de contraintes de stockage. Elles peuvent être présentées comme suit :

- Conteneur d'usage général (les conteneurs dry) : il peut être stocké sur terre, sur un conteneur de même type ou un conteneur vide
- Conteneur vide : il ne peut être stocké que sur un conteneur vide ou sur terre.
- Conteneur à toit ouvert : il peut être stocké sur terre, sur un conteneur à usage général ou vide. En plus, aucun conteneur ne peut être mis au dessus.
- Conteneur plate-forme : C'est un conteneur ayant le toit et un coté ouverts. Il peut être stocké sur terre, sur un conteneur à usage général ou vide. En plus, aucun conteneur ne peut être mis au dessus ou sur tout l'espace du coté ouvert.
- Conteneur citerne : Il ne peut être mis que sur terre ou sur un conteneur citerne.
- Conteneur réfrigéré : Il ne peut être stocké que dans un block réfrigéré.

La formulation mathématique de ce problème est présentée dans la prochaine sous-section.

2. Formulation mathématique

Le problème défini ci-dessus, est formulé dans cette sous-section. Cette formulation tient compte de l'ensemble des hypothèses décrites pour le PSC à un seul type de conteneurs. En outre, le plan de stockage déterminé par les approches doit considérer l'état initial des blocs (emplacements vides) et les contraintes de stockage liés aux nouveaux types.

Les différents paramètres utilisés dans cette formulation sont définis comme suit :

- i : L'indice du conteneur, $i = 1, ..., N_c$
- b : Indice d'un block ; $b = 1, ..., N_{Block}$
- $N_{Block} = N_{stock\_reg} + N_{stock\_refrig}$: Le nombre de blocks disponibles
- N_{stock_reg} : Nombre de blocks standards
- N_{stock_refrig} : Nombre de blocks réfrigérés
- N_c : Nombre total de conteneurs à manipuler.
- d_i : Date de départ du conteneur i
- $Nc_{Etage}\ (j,b)$: Nombre de conteneurs à l'étage j du block b.
- n_1 : Le nombre maximum de conteneurs sur l'axe des X
- n_2 : Le nombre maximum de conteneurs sur l'axe des Y
- n_3 : Le nombre maximum de conteneurs sur l'axe des Z (nombre d'étages)
- N_D : Le nombre de types de conteneurs
- $N_c(D)$: Le nombre de conteneurs de type D, Où D = 1, …, 6. dénote le type de conteneur, Tel que :

$$D = \begin{cases} 1 & \text{Si c'est un conteneur à usage général} \\ 2 & \text{Si c'est un conteneur vide} \\ 3 & \text{Si c'est un conteneur à toit ouvert} \\ 4 & \text{Si c'est un conteneur plate-forme} \\ 5 & \text{Si c'est un conteneur citerne} \\ 6 & \text{Si c'est un conteneur réfrigéré} \end{cases}$$

- $C_{i,D}\ (x, y, z, b)$, la variable de décision pour ce problème, $x \in [1,..,n_1]$, $y \in [1,..,n_2]$, $z \in [1,..,n_3]$, $b \in [1, \ldots, N_{Bloc}]$:

$$C_{i,D}(x,y,z,b) = \begin{cases} 1 & \text{Si le conteneur i de type D est à la position } (x,y,z) \text{ du bloc b} \\ 0 & \text{Sinon} \end{cases} \quad [3.6]$$

Ainsi, le problème peut être formulé de la façon suivante :

$$\underset{D=1}{\overset{N_D}{\sum}} \underset{i=1}{\overset{Nc(D)}{\sum}} \underset{b=1}{\overset{N_{Bloc}}{\sum}} m_{i,b} C_{i,D}(x,y,z,b) \quad [3.7]$$

Sous les contraintes :

$$Nc_{Etage}(j,b) \geq Nc_{Etage}((j+1),b), \text{ pour } j = 1...n_3 - 1,\ b = 1...N_{Bloc} \quad [3.8]$$

$$C_{i,D}(x,y,z,b) - C_{i,D}(x,y,z,b) \geq 0, \forall i \in [1..N_c(D)], \forall D \in [1..N_D] \quad [3.9]$$

$$C_{i,r}(x,y,z,b) - C_{j,D}(x,y,z+1,b) = 1, \forall i \in [1..N_c(r)],\ \forall r = 3,4, \forall j \in [1,...Nc(D)] \quad [3.10]$$

$$C_{i,4}(x,y,z,b) - \sum_{m=x+1}^{n3} C_{j,r}(m,y,z,b) = 1, \forall i \in [1..Nc(4)], \forall j \in [1..Nc(D)], \forall r \in [1,..N_D] \quad [3.11]$$

$$C_{i,5}(x,y,z,b) - C_{j,r}(x,y,z+1,b) = 1, \forall i \in [1..N_c(5)], \forall j \in [1..N_c(r)], \forall r \in \{1,2,3,4,6\} \quad [3.12]$$

$$C_{i,6}(x,y,z,b) = \begin{cases} 1, \text{ Si le le bloc est réfrigéré } (b \in [1,...,N_{stock\_refrig}] \\ 0, \text{ Sinon} \end{cases} \quad [3.13]$$

Où

- $m_{i,b}$: Le nombre minimum de conteneurs manipulés pour retirer le conteneur *i*, placé dans le block *b*.

Dans cette formulation :

- Les contraintes 3.8 et 3.9 assurent que le nombre de conteneurs dans un étage d'un bloc *b* est au plus égal au nombre de conteneurs placés à l'étage directement en dessous.
- La contrainte 3.10 veille au fait qu'aucun conteneur ne peut être placé au dessus d'un conteneur à toit ouvert ou d'un conteneur plate-forme.
- La contrainte 3.11 assure qu'aucun conteneur ne peut être placé sur tout l'espace du coté ouvert d'un conteneur plate forme.
- La contrainte 3.12 vérifie que sur un conteneur citerne on ne peut placer qu'un conteneur de même type.
- La contrainte 3.13 concerne les conteneurs réfrigérés qui ne doivent être placés que dans un block de stockage réfrigéré.

Dans cette thèse, deux approches sont développées pour le PSC à différents types de conteneurs ; un algorithme génétique et une recherche harmonique. L'adaptation de ces deux techniques est détaillée dans les deux sections suivantes.

V. Adaptation de l'algorithme génétique au PSC à plusieurs types de conteneurs

L'algorithme génétique est une approche évolutionnaire qui a prouvé son efficacité pour plusieurs problèmes. Une adaptation de cet algorithme est détaillée dans cette section.

Initialement, N solutions seront créées pour former la population initiale. Puis, deux parents sont sélectionnés selon la méthode de la roulette et des solutions nouvelles N générées en utilisant l'opérateur de croisement à un point et un opérateur de mutation. Les nouvelles solutions sont ajoutées à la population initiale de l'itération en cours pour former une population intermédiaire notée P_{inter}. Les opérations de sélection, croisement et mutation sont répétées jusqu'à avoir *2N* chromosomes dans la population P_{inter}. Par la suite, P_{inter} sera triée dans l'ordre croissant selon la fonction fitness de ses solutions. Les premiers individus *N* de P_{inter} formeront la population initiale pour l'itération suivante. Cette procédure est répétée jusqu'à ce que le critère d'arrêt (nombre d'itérations maximal) soit satisfait. Les différentes étapes de l'AG sont détaillées dans la prochaine sous-section.

1. Description de l'AG pour le PSC à plusieurs types

a. Codage de la solution :

Pour représenter une solution au problème étudié, un plan à quatre dimensions (X, Y, Z, B) est utilisé. Il contient les conteneurs affectés à des positions de coordonnée (X, Y, Z) du block B. Chaque conteneur est identifié par son numéro et son type (i, D). L'affectation des conteneurs doit tenir compte de l'état initial des blocs de stockage et des contraintes définies précédemment.

Un exemple d'état initial de la zone de stockage illustré dans la figure ci-dessous.

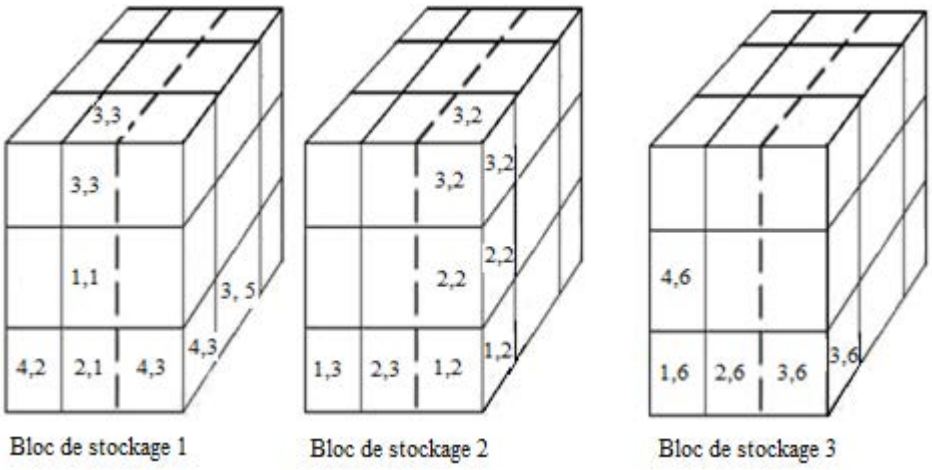


Figure 3.10. Exemple d'état initial

Après l'affectation de 5 nouveaux conteneurs de type général (Dry), l'état de la zone de stockage est illustré dans la figure 3.11

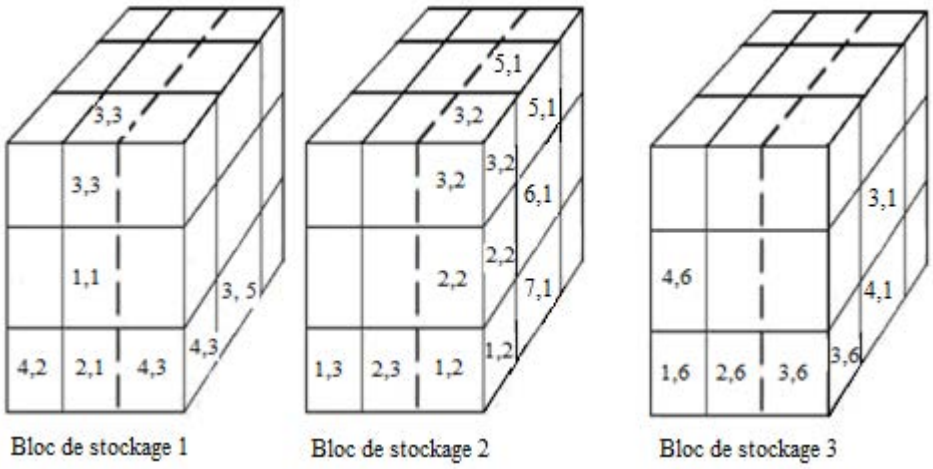


Figure 3.11. Etat initial après affectation des conteneurs

b. Génération de la population initiale

Le principe de création de la population initiale est similaire à celui utilisé pour le PSC à un seul type. Sauf qu'un conteneur ne sera affecté que si l'emplacement disponible, trouvé dans le block sélectionné aléatoirement, vérifie les contraintes relatives à son type. Ceci à fin de garantir la validité de la solution générée.

Algorithme de création de la population initiale
Début
// Cet algorithme contient tous les types
- Consulter l'état initial des blocs de stockage
- Stocker les conteneurs de type réfrigéré dans les emplacements vides des blocs isothermes.
- Stocker les citernes tout en respectant ces contraintes.
- Stocker les conteneurs de type général « Dry » tout en respectant ces contraintes.
- Stocker les conteneurs de type plate-forme tout en respectant ces contraintes.
- Exiger le vide sur les conteneurs plate-forme et à leurs côtés ouvertes
- Stocker les conteneurs de type toit ouvert tout en respectant ces contraintes.
- Exiger le vide sur les conteneurs toit ouvert
- Stocker Les conteneurs de type vide tout en respectant ces contraintes.

Fin

Figure 3.12. Algorithme de création de la population initiale (plusieurs types de conteneurs)

c. Croisement

L'opération de croisement commence par l'affectation de l'état initial de la zone de stockage à chacun des deux enfants (*E1, E2*). Par la suite, deux individus de la population initiale seront sélectionnés et un plan de croisement à un point formé par trois coordonnées *p-crois-x*, *p-crois-y* et *p-crois-z* choisis aléatoirement des axes *X*, *Y* et *Z,* est défini.

L'enfant E_1 recevra les gènes (conteneurs) de l'individu I_1 situés dans le plan de croisement et les conteneurs manquant seront affectés selon l'ordre de leur apparition dans l'individu I_2 tout en respectant les contraintes relatives au type de chaque conteneur.

Si l'emplacement du conteneur dans *I2*, est convenable pour la nouvelle solution c'est-à-dire toutes les contraintes de stockage sont respectées, le conteneur est mis dans cette même position. Cette opération peut, parfois, être impossible. Ceci est dû toujours aux contraintes de certains conteneurs qui rendent un grand nombre d'emplacements vides inexploitables. Dans ce cas, nous cherchons la meilleure position adéquate au conteneur pour le stocker. De même, l'enfant E_2 recevra les gènes de l'individu I_2 situés dans le plan de croisement et le reste des conteneurs sera affecté en utilisant la même démarche que pour E1. La figure ci-dessous décrit l'algorithme de croisement.

Algorithme de croisement pour plusieurs types
Début
- Affectation de l'état iniatial du plan de stockage à l'enfant E1
- Sélection de 2 parents (I1, I2) en utilisant la méthode de la roulette
- Choisir au hasard 3 coordonnées de croisement (*p-crois-x*, *p-crois-y* et *p-crois-z*) respectivement dans [0, .., n_1-1], [0,…,n_2-1]et [0,…,n_3-1]
- Placer les conteneurs occupants dans I1, le plan [0, p-crois-x],[0, p-crois-y] et [0, p-crois-z] dans l'enfant E1.
- Déterminer la liste des conteneurs qui ne sont pas encore stockés dans E1 // La taille de la liste est Nc_{rest}
  // Le conteneur qui reste noté r ∈ [1, Nc_{rest}]
- Répéter
  - ✓ Trouver l'emplacement du conteneur r dans I2
  - ✓ Si l'emplacement est convenable pour E1
    Stocker le conteneur r
    Sinon
    Chercher le plus proche emplacement adéquat et stocker le conteneur r
  - ✓ Mise à jour de la liste des conteneurs restant

  Tant que (r ≤ Nc_{rest})

Figure 3. 133. Algorithme de croisement pour plusieurs types de conteneurs

d. Mutation

L'opérateur de mutation adopté dans cette adaptation consiste à permuter dans chaque block, deux conteneurs ayant le même type.

Algorithme de mutation pour plusieurs types de conteneurs
Début
- Choix arbitraire d'un type de conteneur parmi les types qui existe dans la population
- Choix arbitraire de 2 conteneurs de ce type
- Permuter les 2 conteneurs

Fin

Figure 3.14. Algorithme de mutation pour plusieurs types de conteneurs

e. Mise à jour de la population

Les enfants créés après sélection croisement et mutation sont ajoutés à la population initiale sans vérification de leur qualité. Ils peuvent ainsi participer à la création d'autres enfants ce qui permet une meilleure exploration de l'espace de recherche.

Ce processus de sélection, croisement et mutation est itéré jusqu'à avoir une population de *2N* individus. Parmi lesquels, seuls les N meilleurs formeront la population initiale pour l'itération suivante.

f. Critère d'arrêt

Le processus de génération décrit ci-dessus s'arrête lorsqu'il n'y a pas amélioration de la meilleure solution pour un nombre d'itération maximal, N_{iter}, à définir par l'utilisateur.

VI. Adaptation de l'algorithme de recherche harmonique au PSC à plusieurs types

La recherche harmonique est une approche qui a prouvé son efficacité pour résoudre certains problèmes d'optimisation. De plus, elle a enregistré de meilleurs résultats par rapport à l'AG lors de son application pour le PSC à un seul type de conteneurs. Dans cette section, une adaptation de cette approche pour la résolution du PSC à plusieurs types est proposée

1. Description de l'algorithme de RH pour le PSC à plusieurs types

a. Initialisation des paramètres :

Le vecteur X regroupant les variables de décision est composé par l'ensemble des conteneurs i de type D à stocker dans les emplacements vides des blocs de stockage.

$$X = \begin{bmatrix} 1,1 \\ 2,2 \\ . \\ . \\ N_c(1),1 \\ 1,2 \\ 2,2 \\ . \\ . \\ N_c(2),2 \\ . \\ . \\ 1,6 \\ . \\ N_c(6),6 \end{bmatrix} \qquad [3.14]$$

On associe à chaque type de conteneurs (D), un ensemble d'emplacements (X_D) pouvant stocker un conteneur de ce type. L'ensemble des valeurs possibles pour la variable de décision i,D est X_D .

Une mise à jour de la liste des emplacements X_D est effectuée après l'affectation d'un conteneur de type D.

b. Initialisation de la population initiale

La population initiale qui forme la mémoire harmonique (HM) est générée en utilisant la même que celle utilisée pour créer la population initiale dans l'AG.

c. Création d'une nouvelle solution

La même démarche de la création d'une nouvelle solution pour le cas d'un seul type de conteneurs est utilisée pour chaque type de conteneurs tout en vérifiant chaque fois les contraintes de stockage. La figure ci-dessous présente l'algorithme.

Algorithme Création d'une nouvelle solution

Début
// Le nombre de conteneurs à stocker de type D est $N_c(D)$
// Le nombre des types est N_D
Pour (i = 1 à N_D)
n= 0
Répéter
- Choix arbitraire d'un conteneur c de type D
- Choix arbitraire d'une probabilité : P_{HMCR}
- Si $P_{HMCR} < 90\%$
  - ✓ Choix arbitraire d'une solution S de la population initiale
  - ✓ Si l'emplacement du conteneur c dans S est convenable (les contraintes de stockage sont respectés) pour la nouvelle solution
    - Stocker le conteneur c dans cette place
    - Choix arbitraire d'une probabilité : P_{PAR}
    - Si $P_{PAR} < 10\%$
      - Appliquer la procédure d'ajustement
      - Stocker le conteneur c dans la position ajustée
      - n = n+1
- Sinon
  - ✓ Sélectionner aléatoirement un emplacement convenable pour le conteneur c à partir de sa liste des positions
  - ✓ Stocker le conteneur c
  - ✓ n = n+1

Tant que ($n \leq N_c(D)$)
Fin Pour
Fin

Figure 3. 15. Algorithme de création d'une nouvelle solution (différent types de conteneurs)

La nouvelle solution est ajoutée à la HM si elle est meilleure que la plus mauvaise solution déjà trouvée. Les opérations de création d'une nouvelle solution et de mises à jour de la

mémoire harmonique s'arrêtent lorsqu'il n'y a pas d'amélioration de la meilleure solution pour un nombre N_{iter} d'itérations.

Conclusion

Au cours de ce chapitre, le PSC statique à un seul et plusieurs types de conteneurs à été étudié. La formulation mathématique a été détaillée pour chaque cas. Par la suite, l'adaptation des deux méthodes de résolution, l'algorithme génétique et la recherche harmonique, ont été proposées. Dans le prochain chapitre, nous allons traiter l'aspect dynamique du PSC.

Chapitre 4 : Résolution du PSC Dynamique pour plusieurs types de conteneurs

Or, quand on travaille pour demain, et pour l'incertain, on agit avec raison : Car on doit travailler pour l'incertain par la règle des parties qui est démontrée.
[Pascal]

Introduction

Un terminal à conteneurs est un ensemble de systèmes logistiques complexes avec des interactions dynamiques et aléatoires ce qui ne permet pas d'envisager une pré-planification des activités de transport et de manutention des conteneurs au delà de 5-10 min. Ainsi, le contrôle et le traitement des données en temps réel il est indispensable pour assurer une prise de décision efficace.

L'affectation des conteneurs arrivant au port, à leurs emplacements de stockage est l'une de ces décisions. Les informations liées au flux des conteneurs sont souvent incertaines. Les modifications apportées aux données sont dues aux imprécisions des dates d'arrivée et de départ des navires (raisons météorologiques) et des heures d'arrivées des camions des clients (problème de circulation sur les routes). En effet, le problème de stockage de conteneurs statique devient un problème dynamique, ceci fait l'objet de ce chapitre.

Nous décrivons en premier lieu les problèmes dynamiques et ses caractéristiques. Ensuite, nous présentons les principes méthodes utilisées pour la résolution. Puis, nous abordons le PSC dynamique. Nous décrivons le problème, les travaux qui l'ont traité et nous énonçons sa formulation mathématique. Par la suite, nous détaillons les deux approches proposées pour résoudre le PSC dynamique et l'adaptation des méthodes de résolution, l'algorithme génétique et la recherche harmonique, au problème.

I. Les problèmes dynamiques

1. Définition d'un problème dynamique

Un problème d'optimisation dans un environnement dynamique est un problème ayant une fonction objectif (et/ ou) des contraintes modifiables au cours du temps. Ainsi, une solution optimale à un moment donné peut devenir partiellement optimale dans des périodes ultérieures et de nouvelles solutions peuvent apparaître.

Un problème d'optimisation dynamique peut être définit comme suit :

$$\text{Min } f(x, \delta(t)), x = (x_1, \dots, x_{nx}), \delta(t) = (\delta_1(t), \dots, \delta_{n\delta}(t)) \quad [4.1]$$

Sujet à

$$g_m(x, (t)) \leq 0, m = 1, \dots, n_g \quad [4.2]$$

$$h_m(x, (t)) = 0, m = 1, \dots, n_h \quad [4.3]$$

avec :

$\delta(t)$: le vecteur des paramètres de contrôle de la fonction objectif à l'instant t,

$n\delta$: le nombre des paramètres de contrôle,

g_m et h_m décrivent les contraintes d'inégalité et d'égalité, respectivement.

Dans ce problème, l'objectif principal est de trouver et de suivre la trace de :

$$x^*(t) = \min_x f(x, \delta(t)) \quad [4.4]$$

avec x* la solution optimale trouvée à l'instant t.

La tâche principale d'un problème dynamique est de :

- Localiser l'optimum ainsi que de suivre la trace de sa trajectoire de près
- Rechercher de nouveaux optimums qui peuvent apparaître lors du processus d'optimisation.

Pour cela, les algorithmes adoptés doivent être flexibles avec l'éventuel changement de la position de x*(t) ainsi que de la valeur de l'optimum, f(x*(t)).

2. Caractéristiques d'un problème dynamique

L'optimisation dynamique constitue un domaine de recherche très important comme une partie significative des problèmes du monde réel qui sont des problèmes soumis à des environnements dynamiques et incertains, à savoir les applications de planification où de nouvelles tâches peuvent être insérées à tout moment comme dans les problèmes des tournées de véhicules où de nouveaux clients peuvent être insérés dans les routes, les problèmes de Bin packing où les dates d'arrivées et de départ des cubes sont souvent aléatoires,…

L'incertitude dans un environnement est du à plusieurs causes :

- Bruit : L'évaluation de fitness peut être soumis à du bruit.

- Perturbation : Les variables de décision peuvent être soumises à des perturbations ou des changements après la détermination de la solution optimale.
- Fitness approximative : s'il n'existe pas une fonction analytique, les fonctions fitness sont souvent approchées, basées sur des données provenant d'expériences ou de simulations.
- Fonction fitness qui varie et qui dépend du temps.

Un problème dynamique est caractérisé par :

- La vitesse de changement : elle dépend de l'évolution de l'environnement, rapide ou lent.
- La gravité des changements : elle dépend de la fréquence des changements (réduite, grande)
- La périodicité de changement : elle est dite cyclique quand l'environnement revient à certains états déjà observé dans le passé. Elle est aléatoire lorsque les modifications se présentent d'une manière aléatoire.

Les particularités d'un problème d'optimisation dynamique sont déterminées par la représentation du problème, les caractéristiques de l'environnement et la qualité des mesures utilisées pour évaluer la performance de l'algorithme.

Dans un système dynamique, la qualité de la solution trouvée sera un critère moins déterminant que pour les méthodes d'optimisation en environnement stationnaire. Par contre, la robustesse de ces méthodes sera évaluée par rapport à leurs capacités à :

- détecter un changement,
- fournir une réponse appropriée. [Berro, 2001]

3. Les principales méthodes de résolution d'un problème dynamique

L'optimisation dynamique est un domaine récent qui consiste à trouver la meilleure solution et les plus adaptée pour un environnement incertain et qui change au cours du temps.

Les algorithmes de résolution se confrontent à une grande difficulté dans l''identification des Optimum dynamiques. Ainsi, les algorithmes doivent trouver un équilibre entre l'exploitation et l'exploration de l'espace de recherche.

Parmi les méthodes connues figurent Monte-Carlo, le Simplexe dynamique, les heuristiques simples, les heuristiques dynamiques et les algorithmes évolutionnaires dynamiques.

a. Le simplexe dynamique

L'algorithme simplexe dynamique [Xiong et Jutan, 2003] est largement inspiré du simplexe de Nelder & Mead. Un simplexe SN est un polytope convexe avec N + 1 sommets $\{x_j\}_{j=1}^{N+1}$ dans un espace à N dimensions dans R^N.

Le simplexe de Nelder & Mead est un algorithme itératif, qui commence à partir d'un simplexe initial, puis teste si la fonction objectif a une meilleure valeur sur un sommet voisin. [Tfaili, 2007]

Une itération k commence par ordonnancer et étiqueter les sommets du simplexe, de telle manière que :

$$f_1^{(k)} \geq f_2^{(k)} \geq \cdots \geq f_{N+1}^{(k)} \qquad [4.5]$$

où $f_j^{(k)} = f_j^{(k)}\left(x_j^{(k)}\right)$ est la valeur de la fonction objectif en $x_j^{(k)}$.

b. Les heuristiques simples :

Parmi les heuristiques simples, nous pouvons présentés celles utilisées la résolution du problème de Bin Packing dynamique. C'est un ensemble de règle peuvent être utilisé pour décider sur l'emplacement de chaque élément arrivant.

Les heuristiques les plus utilisées sont :

- Worst- Fit (BP-WF) : Placez les éléments dans l'ordre dans lequel ils arrivent. Placez le nouvel élément dans la boite qui aura le plus d'espace vide après le stockage de l'élément dedans. S'il n'y pas d'espace suffisant dans une boite, il utilise une nouvelle boite.
- Best Fit (BP-BF) : Placez les éléments dans l'ordre dans lequel ils arrivent. Placez le nouvel élément dans la boite qui va laisser le moins d'espace vide après le stockage de l'élément dedans.
- First-Fit (BP-FF) : Placez les éléments dans l'ordre dans lequel ils arrivent. Placez le nouvel objet dans la boite adéquate ayant le plus petit numéro.
- First Fit Decreasing (BP-FFD) : Trier les objets dans un ordre décroissant. Placez le nouvel élément dans la boite adéquate ayant le plus petit numéro
- Best Fit Decreasing (BP-BFD) : Trier les articles dans un ordre décroissant. Placez le nouvel élément dans la boite qui va laisser le moins d'espace vide après le stockage de l'élément dedans.

c. Les heuristiques dynamiques :

Les heuristiques dynamiques peuvent être regroupées en deux catégories :

- Mode immédiat
- Mode batch

Dans le mode immédiat, une tâche est servie dès qu'elle arrive. En mode batch, les tâches sont collectées et ne sont exécutées qu'à des moments bien définis ou quand un ensemble de conditions seront satisfaites. Il existe cinq heuristiques pour le mode immédiat et cinq autres pour le mode Batch.

c.1. Heuristiques en mode immédiat :

Dans le mode immédiat, seule la nouvelle tâche ou événement est à considérer. Ainsi, le comportement de l'heuristique est fortement influencé par l'ordre dans lequel les tâches arrivent. Parmi les méthodes utilisées pour la gestion des tâches dynamiques nous pouvons citer :

- *Délai d'exécution minimale :* Le principe d'affectation de chaque tâche se base sur la minimisation du temps d'exécution de la tâche indépendamment du nombre de tâches en attente.
- *Temps d'achèvement minimal :* Pour chaque tâche, l'algorithme identifie tout d'abord l'ensemble des solutions et identifie la solution ayant le temps d'achèvement minimal.
- *La robustesse maximale :* La tâche est affectée à la solution ayant la robustesse maximale.

c. 2. Les heuristiques pseudo-batch :

Les heuristiques pseudo-batch utilisent deux sous-heuristiques, une pour affecter la tâche qui arrive et l'autre pour assigner les tâches qui sont en attente.

La distribution initiale est effectuée en utilisant l'heuristique « Temps d'achèvement minimale ». L'affectation des tâches qui sont attente est gérée par les heuristiques suivantes :

- *Minimum temps d'achèvement :* Elle consiste à déterminer pour chaque tâche la solution qui fournit le temps de réalisation minimal. A partir de ces paires de tâche / solution, la paire qui donne le temps global d'achèvement minimal est sélectionnée et la tâche utilise cette solution.
- *Robustesse maximale :* Elle consiste à déterminer pour chaque tâche la solution qui fournit le temps de d'achèvement minimal. A partir de ces paires de tâche /

solution, la paire qui fournit la robustesse maximale est sélectionné et la tâche utilise cette solution.

- *Somme pondérée maximale :* Une fonction d'objectif est définie permettant de minimiser le temps d'achèvement et maximiser la robustesse. Pour chaque tâche, la solution qui fournit la valeur maximale de la fonction objectif est déterminée. Parmi cette collection de paires (tâches / solution), la paire qui fournit la valeur maximale de la fonction objectif est sélectionné. [Mehta et col., 2006]

d. La méthode de Monte-Carlo

On appelle méthode de Monte-Carlo toute méthode visant à calculer une valeur numérique, et utilisant des procédés aléatoires, c'est-à-dire des techniques probabilistes. Le nom de ces méthodes fait allusion aux jeux de hasard pratiqués à Monte-Carlo.

La méthode de simulation Monte-Carlo peut être vue comme une méthode d'approximation. Elle se caractérise par l'utilisation du hasard pour résoudre des problème centrés sur un calcul. Elles sont généralement utilisées pour les problèmes numériques ou bien les problèmes de nature probabiliste.

e. Les algorithmes évolutionnaires dynamiques

Les algorithmes évolutionnaires dynamiques sont parmi les modèles existant et prenant en compte des incertitudes de l'environnement.

Les heuristiques imitant les phénomènes naturels rencontrent des problèmes concernant la convergence prématurée dans des environnements dynamiques qui ne leur permettent pas de découvrir le nouvel Optimum, parce que la population perd la diversité dans le processus itératif. La solution la plus simple pour résoudre le problème de convergence prématurée serait de relancer l'algorithme quand un changement dans l'environnement est remarqué, mais cette approche perd toutes les informations acquises par la population lorsque la recherche est relancée. Ainsi, différentes techniques ont été proposées pour résoudre ce problème.

En 2001, Jurgen Branke a présenté les quatre catégories des algorithmes évolutionnaires d'optimisation de problèmes dynamiques. [Branke, 2001]

- Les méthodes réactives
- Les méthodes qui maintiennent la diversité
- Les méthodes qui utilisent une mémoire
- Les méthodes qui utilisent plusieurs populations

e.1. Les méthodes réactives

Elles réagissent au changement. Dès qu'on détecte un changement dans l'environnement. Ces approches effectuent une action extérieure suite à la détection d'un changement de l'optimum. Généralement, cette action est générée pour augmenter la diversité de la population et faciliter le passage au nouvel optimum. Cela va permettre à l'algorithme de retrouver sa capacité de recherche. [Berro, 2001].

Les méthodes réactives utilisent plusieurs techniques pour favoriser la diversité à savoir l'Hypermutation et la Variable Local Search (VLS).

e. 2. Les méthodes préventives de maintien de la diversité

Elles utilisent une bonne population et une répartition étendue des individus dans l'espace, afin de réagir plus efficacement au changement.

Ces méthodes supposent que lors d'une modification de la fonction objectif, la répartition des individus dans l'espace d'état permettra de retrouver plus rapidement le nouvel optimum.

Les méthodes préventives de maintien de la diversité utilisent les outils suivants pour assurer la diversité :

- Random immigrants
- Sharing (heuristique de partage)
- Crowding (heuristique de remplacement)
- Age des individus

e.3 Les méthodes à mémoire

Ces méthodes utilisent une mémoire qui enregistre l'évolution des différentes positions de l'optimum afin de pouvoir les réutiliser un peu plus tard. La mise en place d'une mémoire dans une population peut se faire d'une manière implicite ou de manière explicite.

- *Les approches implicites* utilisent une représentation différente du génotype afin d'inclure l'historique d'évolution de l'optimum dans son génome.
- *Les approches explicites* utilisent une mémoire externe et nécessite la définition de stratégies d'utilisation de cette mémoire (mise à jour et réutilisation des optima stockés). [Berro, 2001].

e. 4. Les méthodes qui utilisent plusieurs populations :

Ces populations sont réparties sur les différents optimums, de manière à renforcer la probabilité de trouver le nouvel optimum.

II. Le problème de stockage des conteneurs dynamiques

Le problème dynamique de stockage des conteneurs peut être défini comme une variante du problème Bin Packing dynamique.

Le Bin Packing dynamique une généralisation du problème classique de Bin Packing. Elle a été introduite par Coffman , Garey et Johnson [Coffman et col., 1983]. Ce problème suppose que les dates d'arrivée et de départ des éléments sont aléatoires et incertaines.

A son arrivée, un objet est affecté à un espace de stockage parmi les endroits inoccupés de longueur égale à la taille de l'élément. Après une période, l'élément peut partir, ce qui rend son espace disponible pour d'autres articles.

1. Présentation du PSC dynamique

La charge de travail dans un terminal pour une période de temps spécifique peut être mesurée par le nombre de conteneurs traités durant cette période. C'est la somme de deux grandeurs : les conteneurs déchargés des navires, stockés provisoirement dans les zones de stockage et les conteneurs destinés à l'import, arrivés dans des camions externes et stockés dans l'espace de stockage.

Selon [Steenken et col., 2004]: Dans les terminaux européennes 30 - 40% des conteneurs destinés à l'export arrivent au terminal manque de données précises pour le navire respectif, le port de destination, le poids du conteneur - des données qui sont nécessaires pour prendre la meilleure décision concernant le stockage des conteneurs . Même après l'arrivée, le navire et le port de décharge peut être modifiée par la ligne maritime. Pour les conteneurs destinés à l'import, déchargés des navires, la situation est encore pire : dans moins de 10-15% de cas, le mode de transport terrestre est connu au moment de décharger un navire.

La figure ci-dessous montre la répartition des temps de séjour des conteneurs d'exportation et d'importation dans un terminal à conteneurs réel.

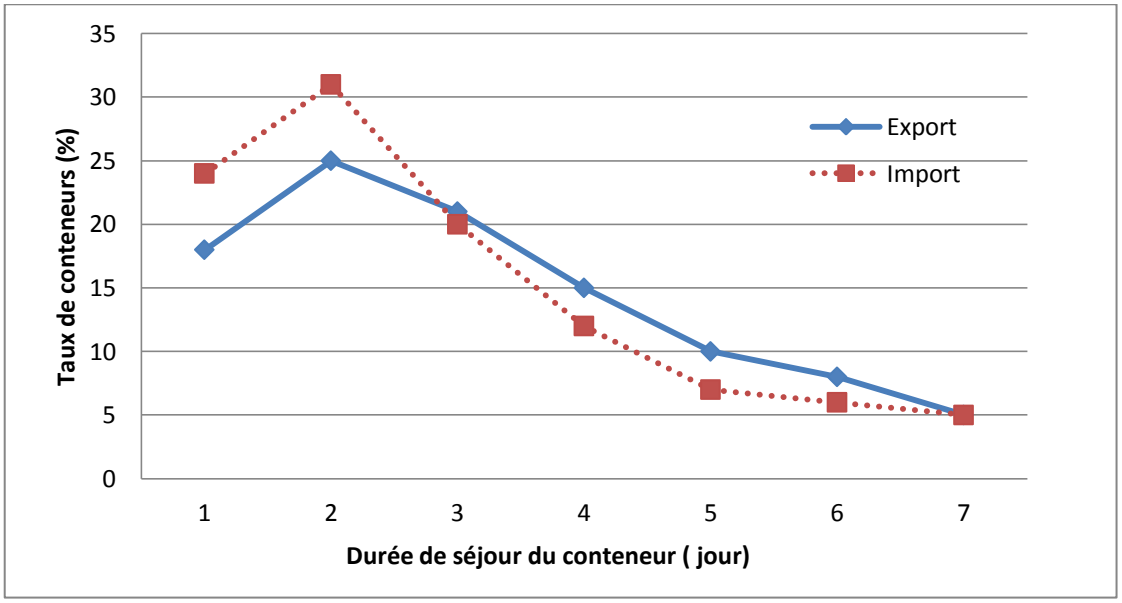


Figure 4. 1. Un exemple de distribution des temps de séjour des conteneurs d'export et d'import [Park et col., 2011]

Les incertitudes sont due généralement à :

- Des informations incomplètes sur l'état initial (poids, destination,…)
- La durée de séjour des conteneurs est inconnue à l'avance. Ainsi, les dates d'arrivée et de départ des navires dépendent des incertitudes météorologiques, l'arrivée de camions des clients aux portes du terminal est soumise à des conditions incertaines de circulation sur les routes.
- L'arrivée des conteneurs au port est parfois inattendue (un navire qui tombe et nécessite un déchargement)
- Les résultats incertains des opérations portuaires
- Des événements non-déterministes exogènes.

Une évaluation correcte de la politique de stockage des conteneurs n'est possible qu'après le départ de tous ou au moins une partie importante des conteneurs empilés parce que nous devons mesurer l'efficacité opérationnelle, non seulement au moment du stockage, mais aussi au moment de la récupération ultérieure.

Bien que la plupart des conteneurs sont récupérés trois ou quatre jours après le stockage, certains conteneurs peuvent rester plus qu'une semaine. L'environnement opérationnel dans un terminal à conteneurs change au cours du temps. Le nombre, la configuration, la répartition, le temps de séjour des conteneurs stockés dans le port changent continuellement.

En effet, la politique d'empilage doit être capable de faire face à ces changements environnementaux.

2. Les méthodes de résolution du PSC dynamique

Les planificateurs modernes ont développé des techniques qui permettent de trouver des plans de stockage efficaces lorsque de nombreuses sources d'incertitude se présentent, mais ces plans nécessitent beaucoup de temps (Blythe J. 1999).

D'autres chercheurs ont utilisé les algorithmes de recherche en ligne « on line » pour résoudre le problème de stockage dynamique. Cette méthode consiste à résoudre le problème « en ligne » c'est-à-dire qu'elle traite toute nouvelle demande ou modification au moment où elle apparaît.

Dans [Park et col., 2011], les chercheurs proposent un algorithme de recherche en ligne qui ajuste dynamiquement la politique de stockage des conteneurs représenté par un vecteur de somme pondéré des valeurs d'évaluation des critères (la catégorie du conteneur, la hauteur de

la pile, la répartition de la charge de l'empilement des grues). Ce travail se concentre sur le problème de la détermination des positions de stockage des conteneurs d'un terminal à conteneurs automatisés (ACT).

L'algorithme proposé permet de créer et d'évaluer plusieurs stratégies de stockage en parallèle dans un intervalle de temps relativement court. Par ailleurs, en raison de la nature dynamique de cette approche, elle peut adapter la politique de l'empilement pour le changement de l'environnement opérationnel.

Notons que le travail antérieur (Kim et col., 2007) a déjà fourni quelques résultats préliminaires sur l'ajustement dynamique des politiques d'empilement, mais l'article [Park et col., 2011] considère plus de facteurs pour déterminer la position d'empilage et fournit une analyse plus compréhensive

Une autre alternative consiste à utiliser les algorithmes de réarrangement (Remarshalling). Ces méthodes consistent à modifier l'organisation des conteneurs dans un bloc pendant le temps d'inactivité des grues et ceci pour améliorer la productivité de l'opération de chargement en réduisant les remaniements et la distance de déplacement des grues.

Pour atteindre une meilleure performance, les opérateurs de terminaux peuvent réorganiser les piles de conteneurs dans un tel ordre qu'il s'adapte à la séquence de chargement.

L'opération de réarrangement est une bonne solution pour faire face aux problèmes encourus par les conteneurs empilés de façon inappropriée. Néanmoins, elle prend des coûts et elle doit être effectuée pendant le temps d'inactivité qui est généralement court et par la suite, on ne peut pas réarranger tous les conteneurs.

Lorsque les conteneurs sont chargés dan le navire, ils sont stockés à bord selon le plan d'arrimage, qui spécifie l'emplacement de chaque conteneur sur le navire. Les plans d'arrimage sont générés quelques heures avant que l'opération de chargement commence.

Lee et Chao proposent une méthode pour réarranger les conteneurs stockés dans les blocs et qui sont destines à l'export afin d'accélérer l'opération de chargement des navires. Compte tenu de la configuration initiale d'une baie de conteneurs, le programme détermine un plan de déplacement des conteneurs dans la baie, afin d'éliminer les mouvements des manipulations des conteneurs lors de l'opération de réarrangement et de chargement ultérieur. L'heuristique proposée consiste en un processus de recherche de voisinage, un modèle de programmation entier, et trois sous-programmes. Les résultats de tests démontrent la performance de l'approche. [Lee and Chao, 2009]

Pour maintenir l'équilibre et la stabilité d'un navire, les conteneurs les plus lourds sont chargés avant les plus légères. Pour parvenir à un chargement efficace, les opérateurs de terminaux classent généralement les conteneurs d'exportation entrant selon leur poids. Ils groupent les conteneurs ayant le même poids dans la même pile. Cependant, l'estimation des poids des conteneurs n'est disponible qu'au moment de leurs arrivées au port. Ce qui fait qu'une pile comprend souvent des conteneurs appartenant à des groupes de poids différents. Ce mélange provoque des mouvements de remaniements supplémentaires lors du processus de chargement du navire.

Dans [Kang et col., 2006][a] et [Kang et col., 2006][b] , les chercheurs ont proposés un algorithme du recuit simulé pour dériver une stratégie plus efficace d'empilage des conteneurs entrants et ayant de poids incertain. Les résultats de simulation ont montré la performance de cette approche. En outre, des améliorations peuvent être accordées à cette méthode en présence d'une meilleure estimation du poids et en appliquant le processus d'apprentissage.

La productivité d'un terminal à conteneurs est très dépendante de l'efficacité de chargement des conteneurs sur les navires.

Dans le papier [Park et col., 2009], les chercheurs proposent des algorithmes de coopération évolutive pour dériver un plan de réarrangement dans un terminal à conteneurs automatisés. La nouvelle approche décompose le problème en deux sous-problèmes: l'un pour déterminer où déplacer les conteneurs et l'autre pour déterminer la priorité du mouvement. .

Jiang et ses collègues présentent un modèle de simulation pour imiter de façon dynamique les opérations de stockage de conteneurs, leurs remaniements et leurs récupérations.

L'animation du modèle permet à l'utilisateur d'évaluer et comparer en temps réel les différentes règles ou algorithmes de remaniement. Ainsi, l'emplacement de chaque conteneur entrant et chaque conteneur déplacé sont déterminé par l'algorithme choisi ainsi que les résultats des opérations sont affichées en temps réel sur l'interface visuelle.

STRIPS-like est une représentation pour des problèmes de planification discrète. Dans [Galuszka, 2011], cette représentation a été adaptée pour modéliser la planification des mouvements de l'équipement « Reach staker » pour accomplir l'opération de chargement et déchargement des conteneurs dans un terminal à Pologne.

Pour remédier à l'incertitude de l'environnement étudié, due généralement à l'imprécision des dates d'arrivée et de départ des conteneurs, le problème de planification est transformé en un programme linéaire. Des contraintes supplémentaires qui représentent les informations incertaines sur l'état initial de la baie de conteneurs et les résultats incertains des actions sont ajoutées.

Meersmans et ses collègues ont étudié le problème de la planification des différents types d'équipements de manutention au terminal à conteneurs automatisé dans un environnement dynamique. Les temps de manutention et l'ordre de la manipulation des conteneurs ne sont pas connus à l'avance. Ils présentent une optimisation heuristique basée sur la recherche en faisceaux et plusieurs règles de répartition. Une mise à jour de la planification est effectuée lorsque de nouvelles informations sont disponibles.

III. Résolution du PSC dynamique à plusieurs types de conteneurs:

Le problème de stockage des conteneurs dynamique est une extension du PSC statique. En fait, on doit satisfaire les arrivées et les départs incertains et imprévus des conteneurs après la préparation et l'exécution d'un plan de stockage. Le planificateur devra alors satisfaire les nouveaux événements en plus des anciennes tout en optimisant les critères choisis et en respectant les contraintes imposées.

D'une façon générale, nous pouvons dire qu'une fonction statique est indépendante du paramètre temps. En revanche, une fonction objectif dynamique dépend du temps. L'optimum global de la fonction que l'on cherche à optimiser peut évoluer avec le temps. En effet, on doit ajouter la notion du temps à la définition du problème du PSC.

Un terminal à conteneurs fonctionne chaque jour (365 jours par an). Nous décomposons la journée en plusieurs intervalles de temps (période) et dans chacun d'eux, nous résolvons le PSC d'une manière statique (comme indiqué dans le chapitre 3). Une période n'est crée qu'à l'arrivée d'un ensemble de conteneurs au port (import ou export) nécessitant leurs stockages provisoires.

Les événements dynamiques (arrivé imprévu des conteneurs, date de départ de conteneur modifiée, …) d'un intervalle de temps sont traitées dans l'intervalle suivant. Pour résoudre l'incertitude des dates de départ des conteneurs et leurs changements, nous enviseageons deux solutions :

- Au début de chaque période, nous utilisons une méthode de réarrangement (remarshalling). Cette méthode consiste à modifier l'organisation des conteneurs de en

tenant compte des nouvelles informations concernant, généralement, leurs dates de départ et la séquence de déchargement.

- Les conteneurs ayant des dates de départ modifiés sont retirés des blocs de stockage ainsi que ceux qui lui sont au dessus. Ils seront ajoutés à l'ensemble des nouveaux conteneurs arrivant au port pour être stockés.

Après la modification de l'état initial des blocs de stockage selon chacune des méthodes décrites ci-dessus, un algorithme de recherche harmonqiue est adopté afin de déterminer un plan de stockage des différents conteneurs. Nous avons opté pour la RH puisqu'elle a prouvé sa performance et sa supériorité par rapport à l'AG pour les cas statique.

1. Formulation mathématique du PSC dynamique

L'objectif du PSC dynamique est de minimiser le nombre de mouvements de remaniements qui se produisent pour modifier ou améliorer l'organisation des conteneurs initiales ainsi que le nombre de mouvements de remaniement des tous les conteneurs existant dans les zones de stockage lors de leurs départs du port.

Les différents paramètres utilisés dans cette formulation sont définis comme suit :

- i : L'indice du conteneur, $i = 1, \ldots, N_c$
- b : Indice d'un block ; $b = 1, \ldots, N_{Block}$
- $N_{Block} = N_{stock\_reg} + N_{stock\_refrig}$: Le nombre de blocks disponibles
- d_i : Date de départ du conteneur i
- N_D : Le nombre de types de conteneurs
- $N_c(D)$: Le nombre de conteneurs de type D, Où D = 1, …, 6. dénote le type de conteneur.
- $C_{i,D}(x, y, z, b)$, la variable de décision du problème (voir équation 3.6)
- N_{cm} : le nombre de conteneurs existant dans les blocs (état initial) et ayant des dates de départ modifiées.

L'équation ci-dessous traduit l'objectif.

$$\text{Min } (\sum_{cm=1}^{N_{cm}} m_{cm} + \sum_{D=1}^{N_D} \sum_{i=}^{Nc(D)} \sum_{b=1}^{N_{Bloc}} m_{i,b}\, C_{i,D}(x, y, z, b)) \qquad [4.6]$$

avec:

- m_{cm} : le nombre de mouvements cumulés à la $n^{ème}$ période lorsque des modifications sont apportées à un conteneur cm.

- $m_{i,b}$: le nombre minimum de conteneurs manipulés pour retirer le conteneur i dans le bloc b

Toutes les contraintes énoncées dans la formulation du PSC pour différents types de conteneurs sont à respecter (équation [3.8] à [3.13]).

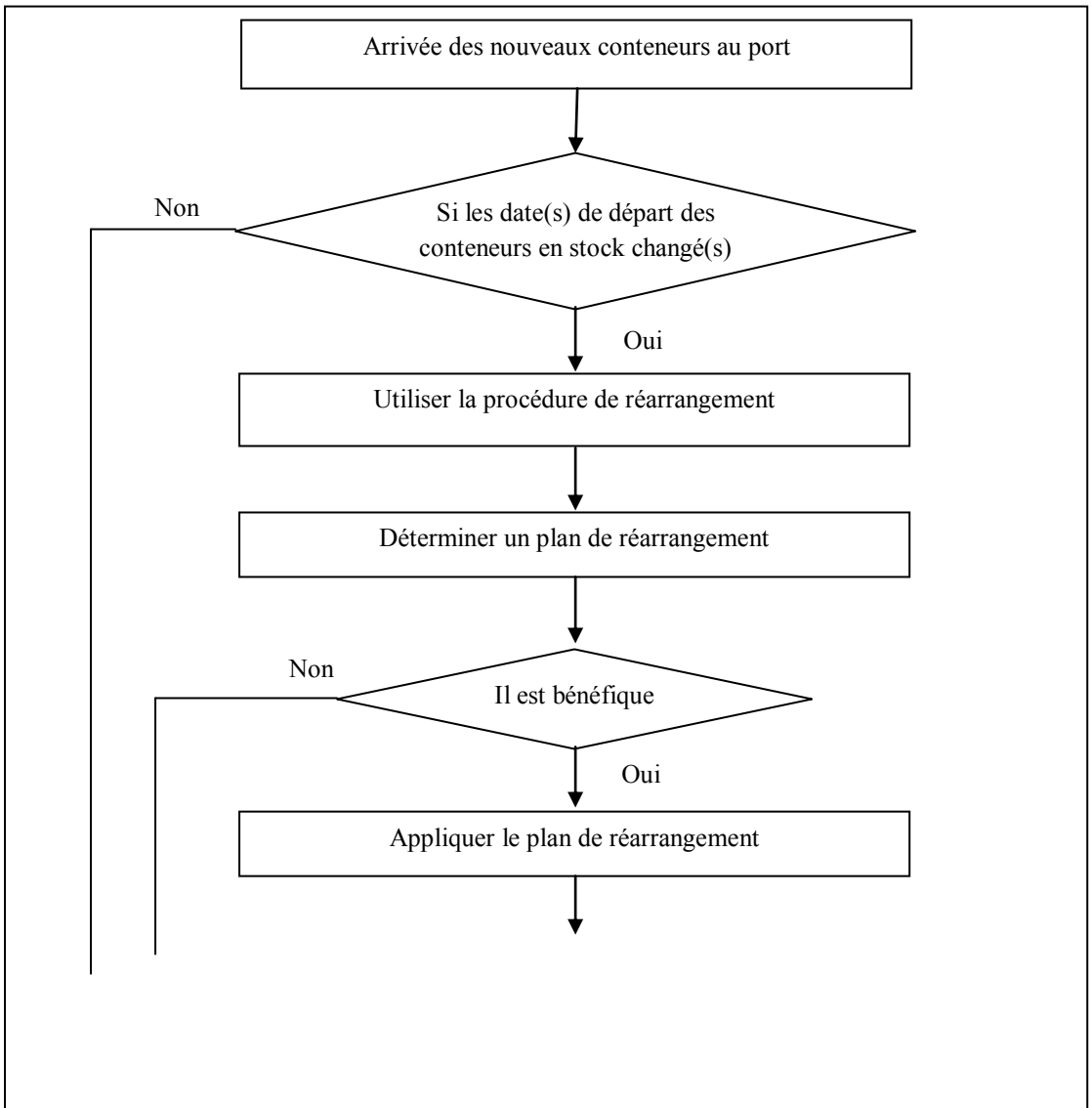
2. 1ère Approche dynamique proposée pour le PSC (plusieurs types)

a. Description de la 1ère approche

La 1ère approche dynamique que nous adoptons pour remédier aux incertitudes liés surtout aux dates de départ des conteneurs consiste à partitionner la journée en un nombre défini de période de travail.

Durant chaque période, une réorganisation du plan de stockage initiale est établie s'il y a des modifications des dates de départ des conteneurs déjà en stock. Ensuite, nous appliquons l'algorithme de la RH pour générer le plan de stockage le plus adéquat des nouveaux conteneurs en considérant la nouvelle situation des blocs (après réarrangement).

Toute modification pour les dates de départs des conteneurs connue pendant la période du travail N sera donc traitée au début de la période suivante (N+1).



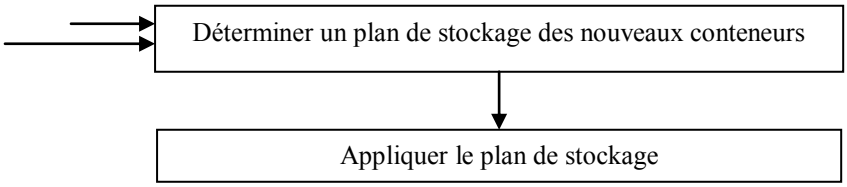


Figure 4. 2. Stratégie de résolution du PSC dynamique

b. L'heuristique de réarrangement

Le processus de réorganisation de l'état initiale des blocks de stockage peut être décrit de la façon suivante :

Pour chaque conteneur dont le délai de départ a changé (noté cm), on cherche, si c'est possible qu'on l'affecte à un nouvel emplacement qui sera bénéfique lors du processus de déchargement de tous les conteneurs. C'est-à-dire, une nouvelle position qui minimisera le nombre de mouvements parasites.

Pour cela, on permute le conteneur cm avec tous les conteneurs de même type et à chaque fois on calcule les 3 paramètres suivants :

- R_{init} : il représente le nombre de mouvements de remaniements des deux piles dans lesquelles se trouvent les 2 conteneurs avant la permutation des 2 conteneurs.
- N_{Permut} : C'est le nombre d'opérations de remaniement nécessaires pour permuter les 2 conteneurs.
- R_{final} = il représente le nombre de mouvements de remaniements des deux piles dans lesquelles se trouvent les 2 conteneurs après la permutation des 2 conteneurs

Puis on calcule le gain, noté G, engendré par cette opération avec :

$$G = (N_{permut} + R_{final}) - R_{init} \qquad [4.7]$$

Ainsi, nous gardons tous les permutations ayant un gain est inférieur à zéro puisque c'est une permutation bénéfique au processus de déchargement (elle minimise le nombre de mouvements parasites) et nous choisissons la meilleure permutation (ayant ($N_{permut} + R_{final}$) minimale).

Si nous avons parcouru toutes les possibilités de permutation et nous n'avons pas trouvé un gain négatif, le conteneur cm garde sa position actuelle.

L'algorithme suivant décrit la démarche de cette procédure de réarrangement.

Algorithme de réarrangement
Début
Pour i=1 à N_{cm}
// cm est le conteneur dont sa date de départ a changé
// Tc : Type du conteneur cm
    Gain = 0
    Pour tout conteneur cm de type Tc faire
        Pour tout conteneur c de type Tc faire
            Rinit = nombre de remaniement pour le déchargement des 2 piles contenant c et cm
            Npermut=nombre d'opérations de remaniement pour la

Figure 4.3. Heuristique de réorganisation des blocs de stockage

Ci-dessous, nous présentons un schéma illustrant le processus de réarrangement. Dans cet exemple, la date de départ de conteneur 9 de type 1 a changé de 4 à 10. En appliquant l'heuristique de réarrangement, une permutation entre ce conteneur et le conteneur 4 de type 1, ayant une date de départ 5 est propice au plan de déchargement. $R_{init} = 10$, $N_{Permut} = 5$, $R_{final} = 3$.

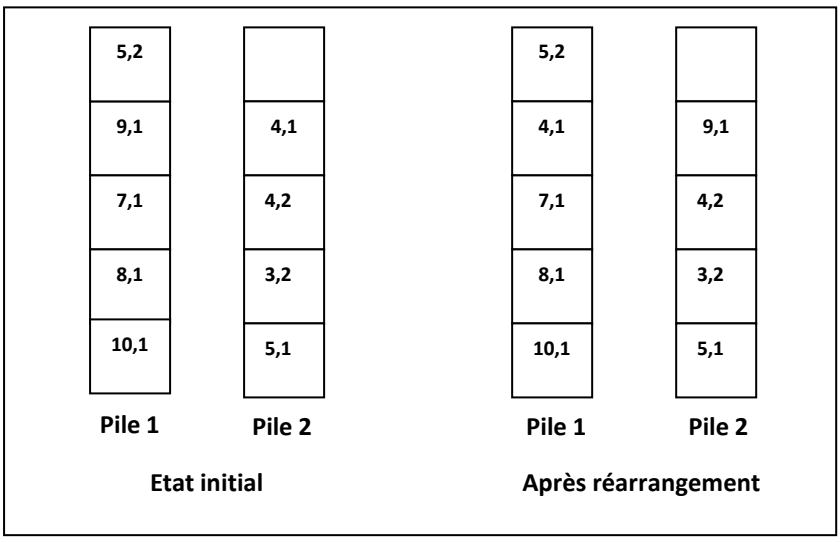


Figure 4.4. Réarrangement des conteneurs

Le plan de réarrangement est ainsi appliqué, un plan de stockage des nouveaux conteneurs est généré par le même algorithme de la recherche harmonique décrit pour le PSC statique plusieurs types.

4. 2ème Approche dynamique proposée pour le PSC (plusieurs types)

a. Description de la 2 ème approche

Cette méthode consiste à retirer le l'état initial tous les conteneurs dont la date a été modifié ainsi que ceux qui leurs sont au dessus. Par la suite l'ensemble de ces contenurs sont ajoutés à la liste des nouveaux conteneurs arrivants. Un algorithme de recherche harmonique est appliqué pour déterminer le nouveau plan de stockage.

b. L'adaptation de la recherche harmonique

Dans cette partie nous présentons l'adaptation de l'algorithme de la recherche harmonique à la résolution PSC dynamique. Cette approche utilise la même représentation des solutions adoptée dans le cas statique. Elle utilise aussi les mêmes variables de décision ainsi que leur liste de choix, la même procédure de création de la mémoire harmonique et la génération d'une nouvelle solution conçus pour le cas statique à différents types de conteneurs. (Voir les paragraphes « Initialisation des paramètres », « Initialisation de la population initiale», « Création d'une nouvelle solution » dans le chapitre 4).

Conclusion

Dans ce chapitre nous avons présenté les problèmes dynamiques, leurs caractéristiques et les méthodes de résolution principales. Nous avons abordé par la suite le problème dynamique de stockage des conteneurs. Nous avons présenté l'état de l'art de ce problème et la formulation mathématique proposée. Ensuite, nous avons détaillé les deux méthodes de résolution que nous avons adoptée pour résoudre les incertitudes des dates de départ des conteneurs ainsi que leurs séquences de déchargement.

Tous ces méthodes ont été élaborés et utilisés dans le but de respecter des contraintes, améliorer les résultats et comme cela sera présenté dans le chapitre suivant dédié à la simulation et aux résultats expérimentaux.

Chapitre 5 : Simulations et Résultats

Il y a si loin de la façon dont on vit à celle dont on devrait vivre,
que celui qui laisse ce qui se fait pour ce qui se devrait faire
apprend plutôt à se détruire qu'µa se préserver.
Niccolo Machiavel

Introduction

Dans ce chapitre nous présentons les différentes étapes d'expérimentation, d'analyse et de simulation faites à travers des exemples pour adapter les méthodes de résolution choisies au problème étudié.

Pour chaque approche proposée dans ce travail, une étude expérimentale est effectuée et des analyses de sensibilité de ses paramètres ont été faites. Une étude comparative des résultats générés par les deux approches proposés, l'algorithme génétique et la recherche harmonique, et l'algorithme LIFO utilisé souvent par les planificateurs au port est présentée pour chaque problème étudié.

I. Le PSC statique : Simulation et résultats

Dans cette partie, nous présentons une étude expérimentale détaillée de l'adaptation de l'algorithme génétique et de la recherche harmonique au PSC statique à un seul et plusieurs types de conteneurs.

1. Le PSC à un seul type de conteneurs

a. L'application de l'algorithme génétique à la résolution du PSC statique à un seul type de conteneur

La réussite de l'adaptation d'une heuristique pour la résolution d'un problème donnée dépend d'un choix judicieux des paramètres de l'algorithme utilisé.

En effet, dans cette section, nous intéressons à l'élaboration d'un ensemble de simulations pour choisir les valeurs les plus adéquates à la taille de la population, la probabilité du croisement, la probabilité de mutation ainsi que le nombre d'itérations.

L'algorithme génétique adopté a été développé en utilisant le langage C et a été exécuté sur une machine a un processeur dual Core de fréquence 1.74GHz chacun et une mémoire RAM de 1GO.

Trois différentes tailles de problème ont été considérées :

- Petite taille : entre 27 et 64 conteneurs
- Taille moyenne : entre 125 et 750 conteneurs
- Grande taille : 1000 conteneurs

Pour chaque taille du problème, *N* solutions sont générées par itération. On suppose que :

- Le nombre de chromosomes (N) générés varie entre 10 et 250.
- n_1, n_2 et n_3 sont identiques pour tous les blocks et sont définis par l'utilisateur.
- Le nombre de conteneurs est à définir par l'utilisateur
- Le nombre de blocs vides est donné par l'utilisateur
- La date de départ de chaque conteneur est définie par l'utilisateur.
- Le critère d'arrêt (nombre d'itérations, noté $N_{itér}$) est compris entre 10 et 300.

a.1. Influence du nombre d'itérations

Le critère d'arrêt est un paramètre très important pour une heuristique. Il est définit généralement en fonction du nombre d'itérations. Dans ce travail, le processus d'optimisation s'arrête lorsque l'algorithme atteint un certain nombre d'itérations sans aucune amélioration au niveau de la fonction objectif définie par l'équation suivante :

$$\mathrm{Min} \sum_{i=1}^{Nc} \sum_{b=1}^{N_{Bloc}} m_i C_i \ (x, y, z, b); \quad \forall x = 1...n_1, \forall y = 1..n_2, \forall z = 1..n_3 \qquad [5.1]$$

Pour un nombre de conteneurs N_c=125 et N=50, nous avons effectué plusieurs tests tout en variant le nombre de générations. La figure 5.1 décrit le résultat de cette analyse.

Nous remarquons que la fonction objectif est stabilisée aux alentours de 37 à partir de 125 itérations.

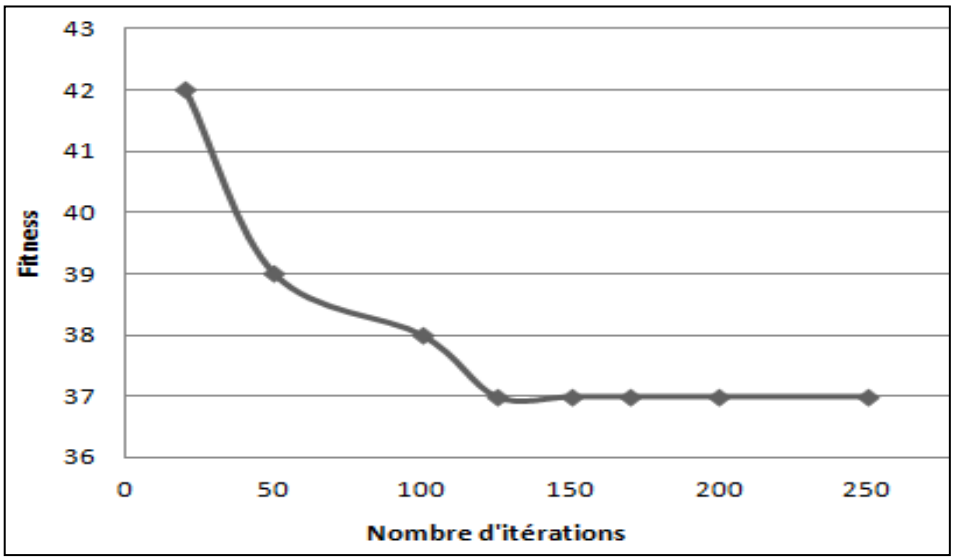


Figure 5. 1. Evolution de la fonction objectif selon Niter

a.2. Influence de la taille de la population

Dans le but d'étudier l'influence de la taille de la population initiale, des expérimentations ont été effectuées pour différentes valeurs de N, avec Nc=140, N_{Block} =6, n_1=n_2=n_3= 3 et $N_{géné}$=125. Les résultats sont affichés dans le tableau suivant :

Tableau 5. 1. Evolution de la fonction objectif selon la taille de la population

N	Fitness
20	58
50	50
75	47
100	41
125	41

Ces résultats montrent que la taille de la population initiale a une influence sur la qualité de la solution finale. En effet, plus la taille de la population initiale est grande, meilleure est la solution générée. D'autre part, l'augmentation de la taille de la population initiale rend le processus d'approximation plus lent.

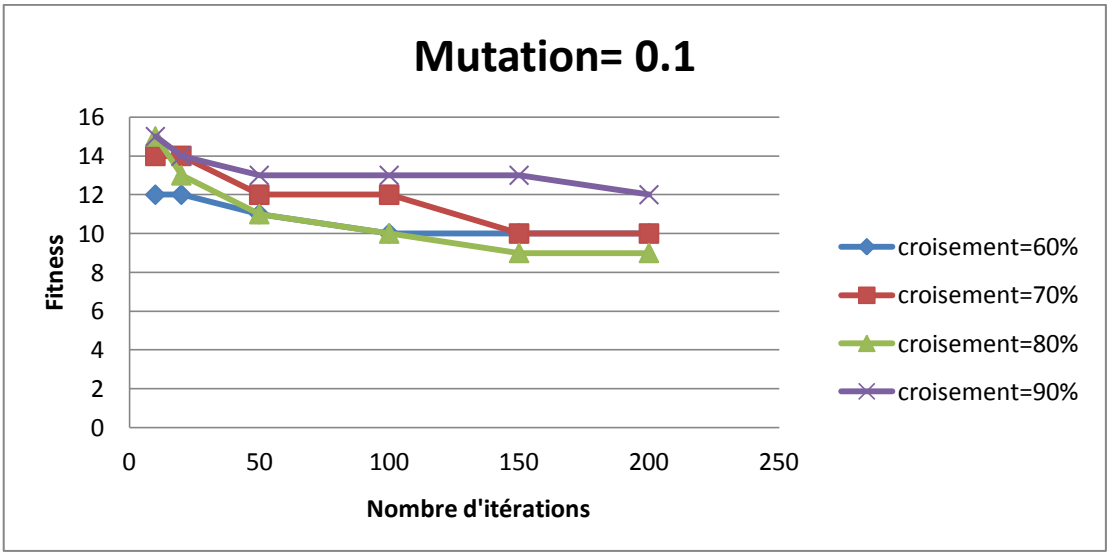
a.3. La probabilité de croisement et la probabilité de la mutation

Les opérateurs génétiques assurent la diversification de la population et l'exploration de l'espace d'état. L'opérateur de croisement manipule la structure des individus, il recompose les gènes. L'opérateur de mutation sert à éviter la convergence prématurée de l'algorithme il apporte la propriété d'ergodicité (atteindre tout les points de l'espace d'état) de parcours d'espace. En effet, il est nécessaire de choisir pour le taux de croisement (P_c) une valeur relativement élevé pour enrichir la diversité des solutions et une faible valeur pour la probabilité de mutation (P_m) pour ne pas tomber dans une recherche aléatoire.

L'étude expérimentale pour cette partie a concerné deux cas :

- Le stockage de 50 conteneurs dans 3 blocs de taille (n1=n2=n3=3)
- Le stockage de 500 conteneurs dans 9 blocs de taille (n1=n2=n3=4)

Pour chaque cas, un ensemble de simulations est établi en variant les valeurs de P_c entre 60% et 90% et P_m entre 10% et 30%. La taille de la population est fixée N= 50 et le nombre d'itération $N_{itér}$ varie entre 10 et 200.



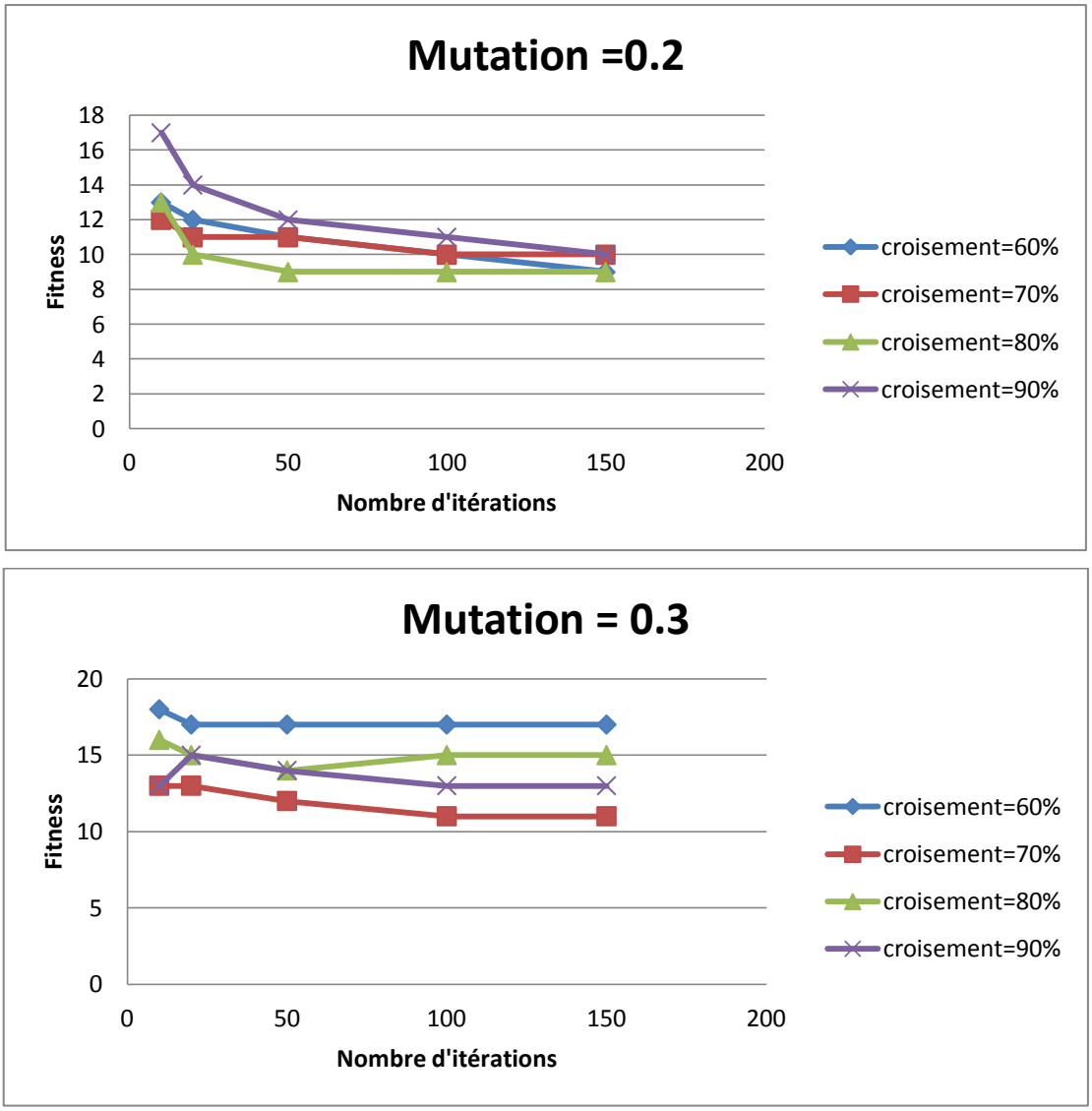
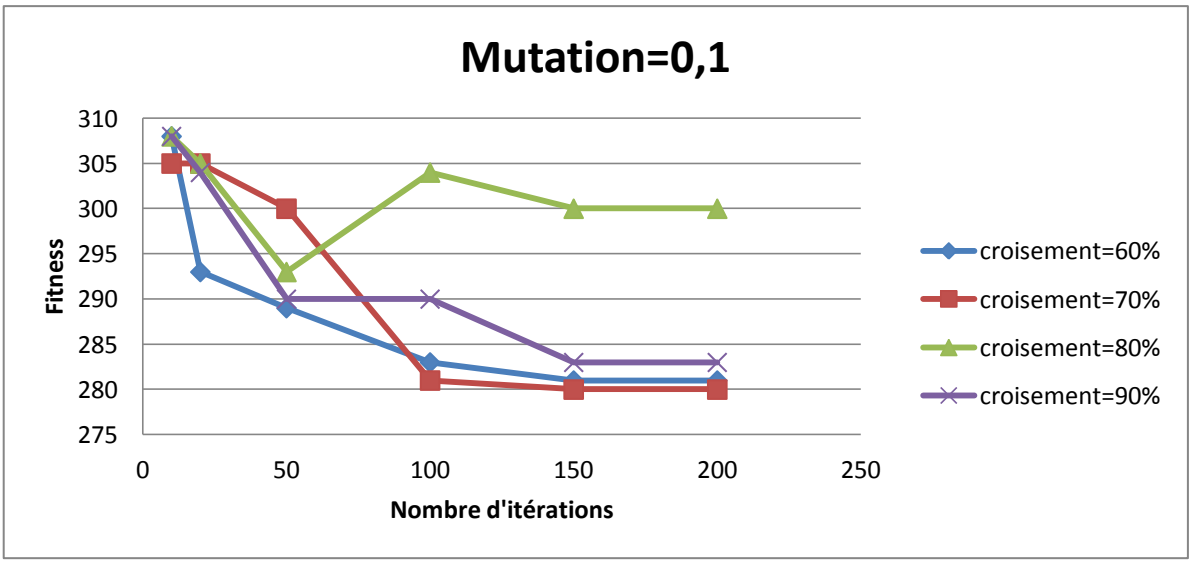


Figure 5. 2. Évolution de la fonction objectif pour différentes valeurs du taux de croisement et du taux de mutation (Nc= 50)



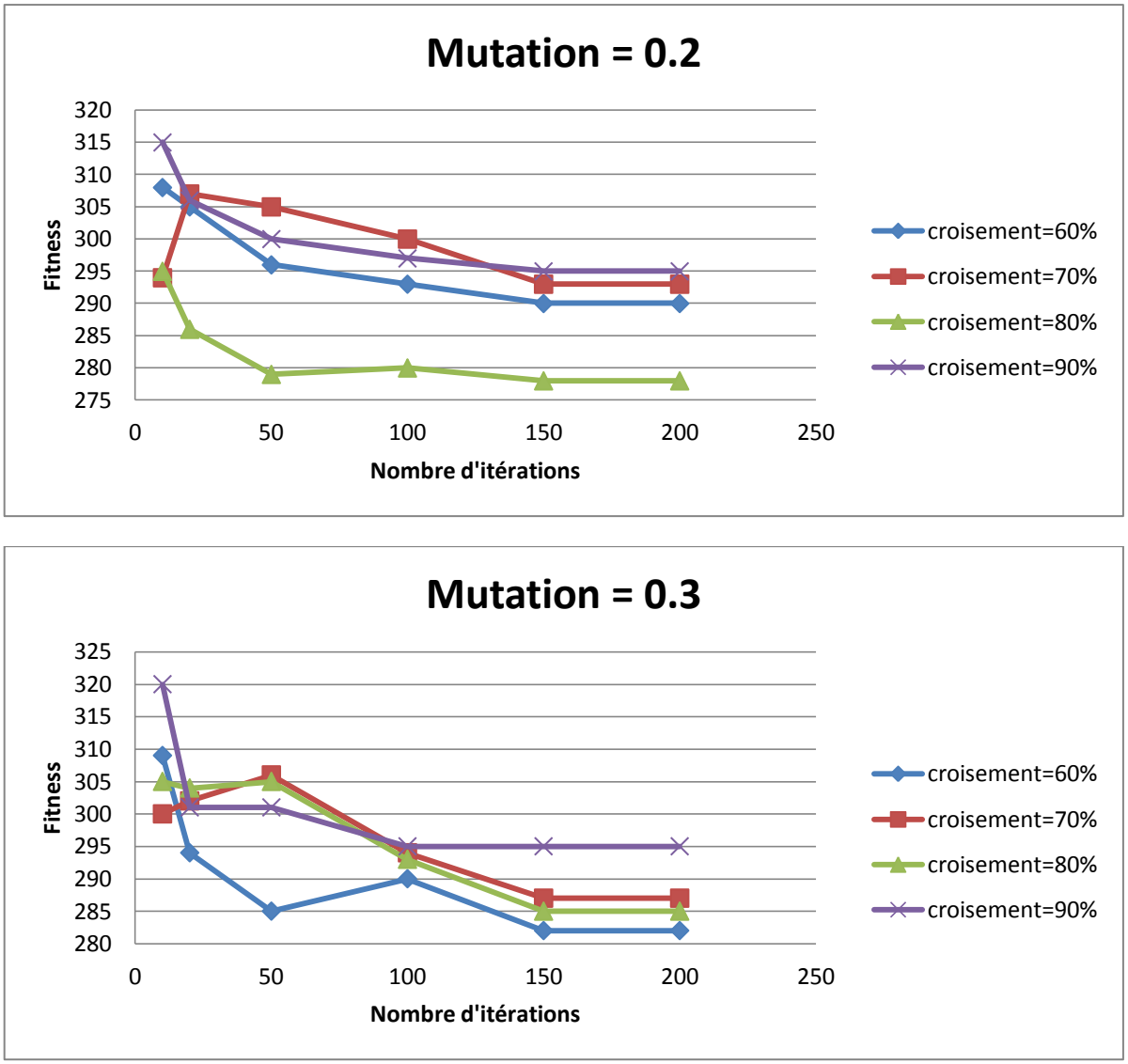


Figure 5. 3. Évolution de la fonction objectif pour différentes valeurs du taux de croisement et du taux de mutation (Nc= 500)

D'après cette étude, nous remarquons que les valeurs du taux de croisement et du taux de mutation les plus appropriées pour notre problème sont respectivement 80% et 20%

a.4. Influence du nombre de conteneurs

Afin d'étudier l'influence du nombre de conteneurs sur la qualité des solutions générées, la taille de la population est fixée à 50 et le nombre de générations à 125. Puis, pour chaque taille du problème, la meilleure valeur de la fonction objectif dans la première itération (notée F_i) et dans la dernière (notée F_f) ainsi que Le taux d'amélioration sont enregistrées.

$$\text{Le taux d'amélioration} = \frac{|F_f - F_i|}{F_f} \qquad [5.2]$$

Les résultats sont donnés par le tableau suivant :

Tableau 5. 2. Influence du nombre de conteneurs sur la fonction objectif

N_c	N_{Block}	n_1, n_2 , n_3	F_i	F_f	Taux d'amélioration (%)
64	3	3	26	14	85,71
125	6	3	61	37	64,86
343	7	4	242	170	42,35
500	10	4	415	301	37,87
729	13	4	567	441	28,57
1000	11	5	950	825	15,15

Les résultats donnés dans le tableau 5.2 indiquent que la valeur de la fonction objectif augmente avec l'augmentation du nombre de conteneurs. Ceci s'explique par la fait que plus le nombre de conteneurs est grand, plus il y a des mouvements de remaniement. D'autre part, le taux d'amélioration du résultat diminue lorsque la taille du problème est grande.

Le nombre de conteneurs influe sur le temps d'exécution. Cette influence est illustrée par la figure suivante :

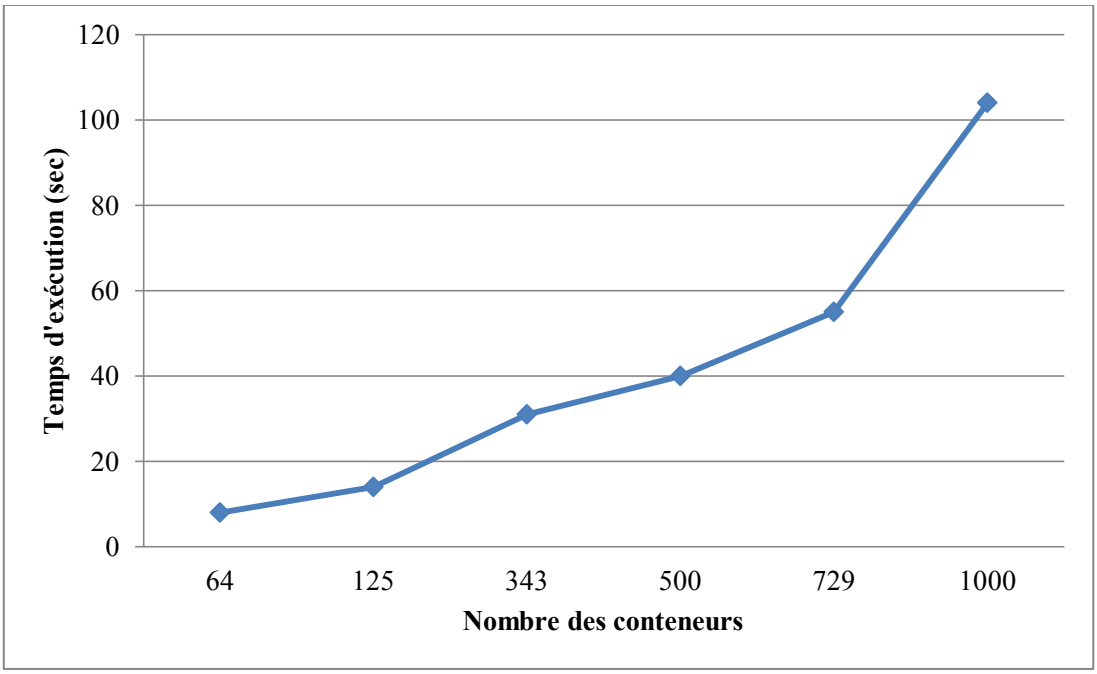


Figure 5. 4. Evolution du temps d'exécution avec N_c

On remarque que le temps d'exécution augmente avec la taille du problème. En effet, pour N_c entre 64 et 1000, le temps d'exécution a augmenté de 8 sec à 104 sec. Ce qui indique que l'algorithme converge rapidement vers la meilleure solution générée.

b. L'application de l'algorithme de la recherche harmonique à la résolution du PSC statique à un seul type de conteneur

b.1. Définition des paramètres du programme

L'approche décrite ci-dessus a été implémentée et exécutée en utilisant le même environnement que celui utilisé avec l'AG.

Les expérimentations sont établies sur trois différentes instances du problème :

- Petite taille : entre 27 et 64 conteneurs
- Taille moyenne : entre 125 et 750 conteneurs
- Grande taille : 1000 conteneurs

Un ensemble de simulations et de tests a été effectué pour chaque instance selon les paramètres suivants :

- La taille de la mémoire harmonique, notée HMS, varie entre 10 et 250.
- n_1, n_2 et n_3 sont identiques pour tous les blocks et sont définis par l'utilisateur.
- Le nombre de conteneurs est à définir par l'utilisateur
- Le nombre de blocs est donné par l'utilisateur
- La date de départ de chaque conteneur est générée de façon aléatoire.
- Le taux de considération de la mémoire harmonique (HMCR) varie entre 60% de 90%
- Le taux d'ajustement (PAR) est entre 10% et 30%.
- Le critère d'arrêt (nombre d'itérations, noté $N_{itér}$) sera entre 20 et 200.

Pour évaluer l'influence de certains paramètres sur les résultats générés, une analyse de sensibilité sur le nombre de conteneurs (N_c), le nombre d'itérations ($N_{itér}$), la taille de la mémoire harmonique (HMS) , la probabilité HMCR et le taux PAR a été établie.

b.2 Influence du nombre d'itérations

Afin d'étudier l'influence du critère d'arrêt, plusieurs simulations ont été effectué pour différentes valeurs du nombre d'itérations ($N_{itér}$) tout en fixant le nombre de conteneurs à 64 et la taille de la population initiale à 50.

Les résultats présentés dans le tableau 5.3 indiquent que la qualité de la solution s'améliore lorsque le nombre de générations augmente mais la fonction objectif se stabilise à partir de l'itération 150.

Tableau 5. 3. Influence du nombre d'itérations

$N_{itér}$	F_i	F_f
20	27	23
50	25	20
100	30	16
150	29	14
175	27	14

200	31	14

b.3.Influence de la taille de la mémoire harmonique

Pour un problème avec 125 conteneurs et un nombre d'itérations fixé à 150 itérations, l'approche a été testée pour différentes tailles de la mémoire harmonique pour évaluer son impact sur la valeur de la fonction objectif. Les résultats sont présentés dans le tableau suivant :

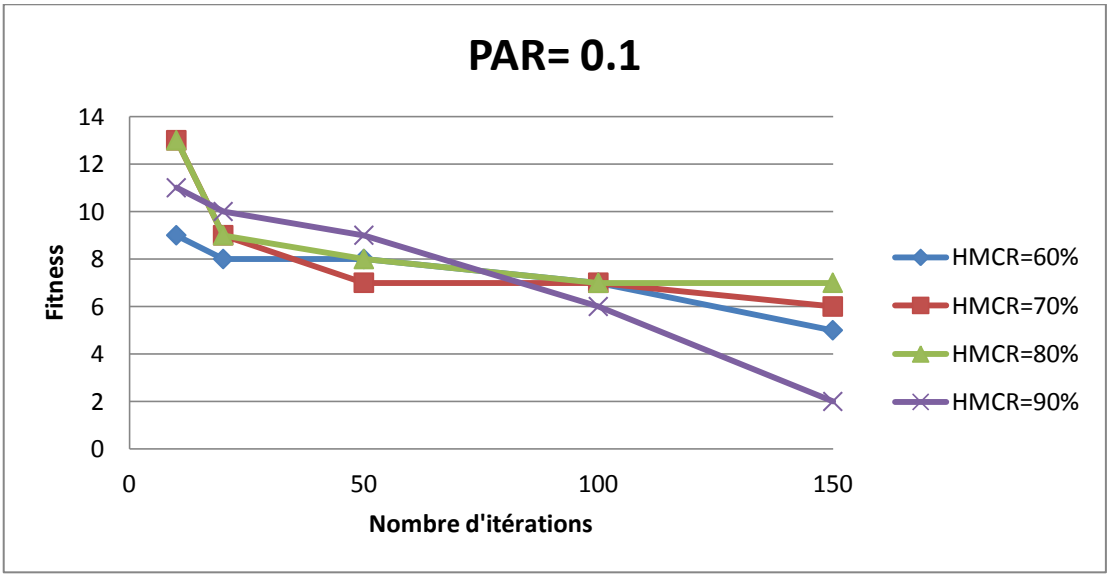
Tableau 5. 4. Influence de la taille de la mémoire harmonique

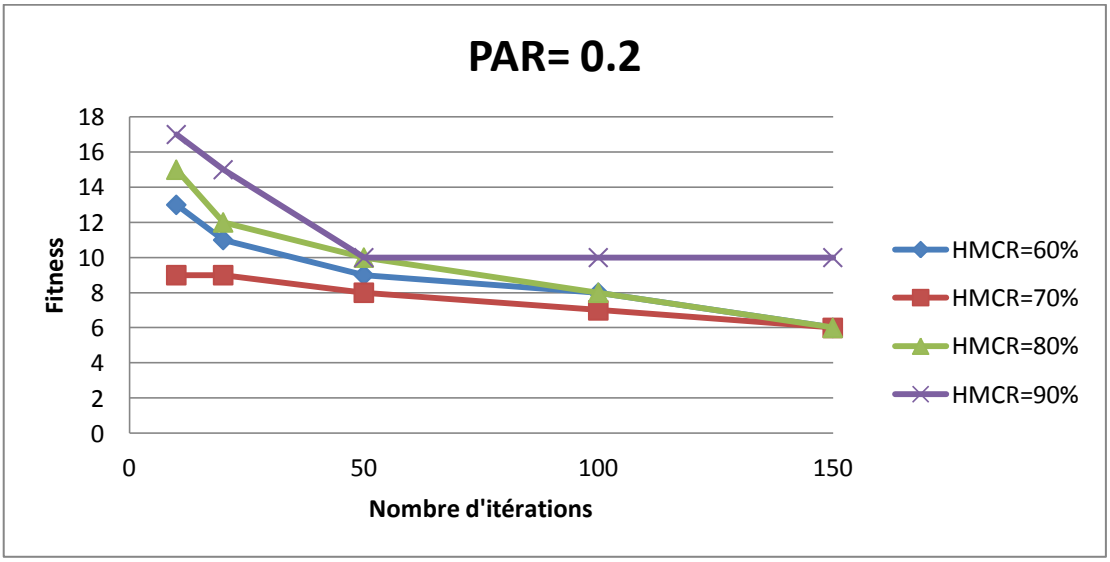
HMS	Fitness
20	38
50	37
75	36
100	33
125	32

Les résultats montrent qu'en augmentant la taille de la mémoire harmonique, la solution devient meilleure. Ceci peut s'expliquer par le fait que lorsque la taille de la mémoire harmonique augmente, le champ d'exploration devient plus grand et mieux exploitable.

b.4.Influence des valeurs du HMCR et du PAR

Les taux de considération de la mémoire harmonique et d'ajustement sont les deux paramètres qu'utilise l'algorithme de la recherche harmonique pour la création d'une nouvelle solution. Pour assurer un choix judicieux des valeurs de HMCR et de PAR et par la suite une meilleure adaptation au problème, nous avons effectué plusieurs simulations en variant les valeurs de ses deux paramètres. Les cas d'étude utilisés sont les mêmes que ceux présentés dans la section a.3.


PAR= 0.1
Fitness
14
12
10
8
6
4
2
0
0
50
100
150
Nombre d'itérations
HMCR=60%
HMCR=70%
HMCR=80%
HMCR=90%


PAR= 0.2
Fitness
18
16
14
12
10
8
6
4
2
0
0
50
100
150
Nombre d'itérations
HMCR=60%
HMCR=70%
HMCR=80%
HMCR=90%

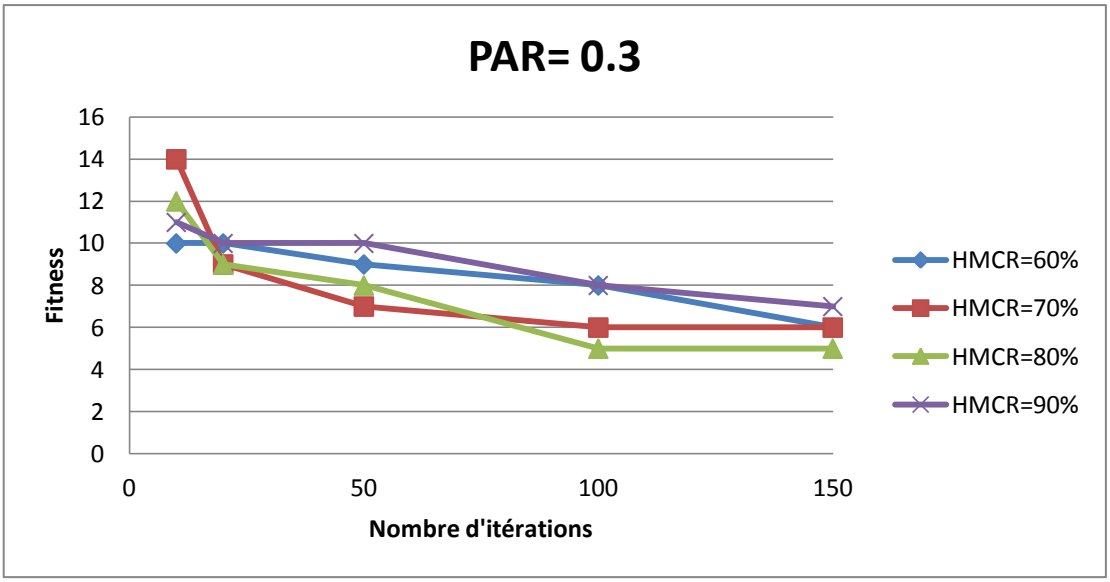
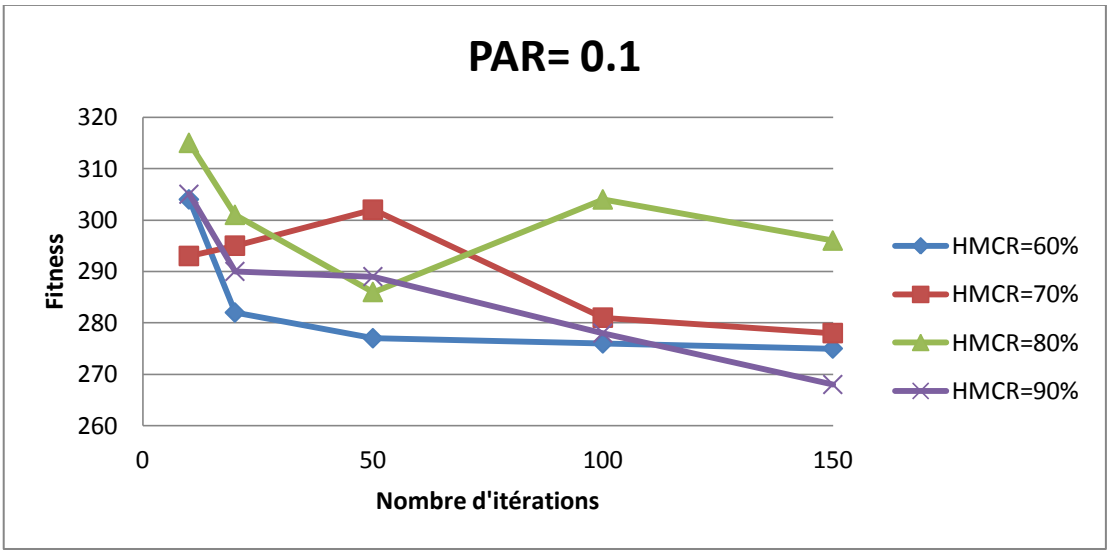
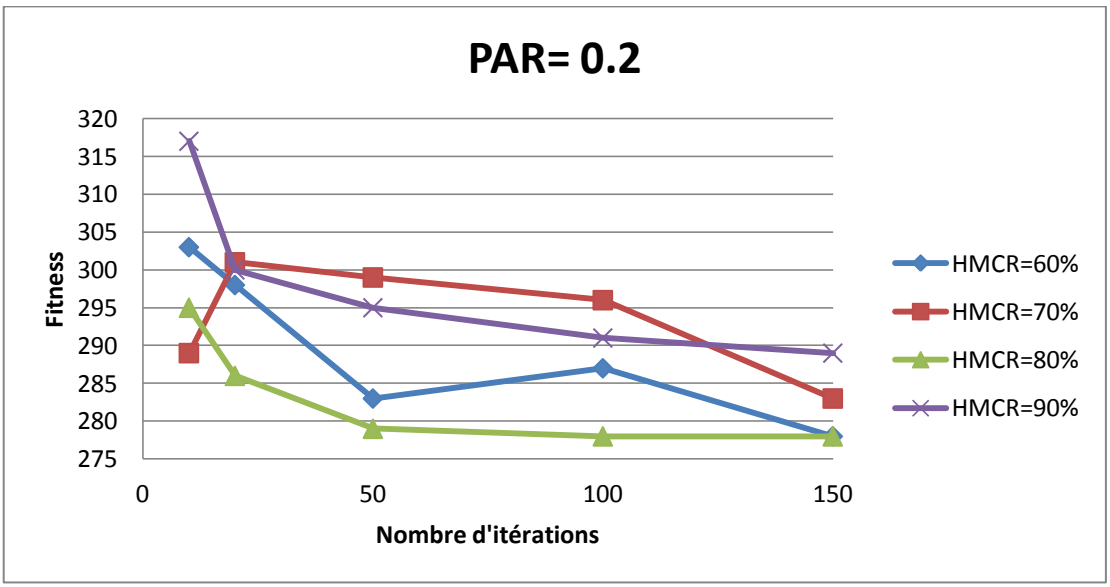


Figure 5. 5. Évolution de la fonction objectif pour différentes valeurs du HMCR et du PAR (Nc= 50)





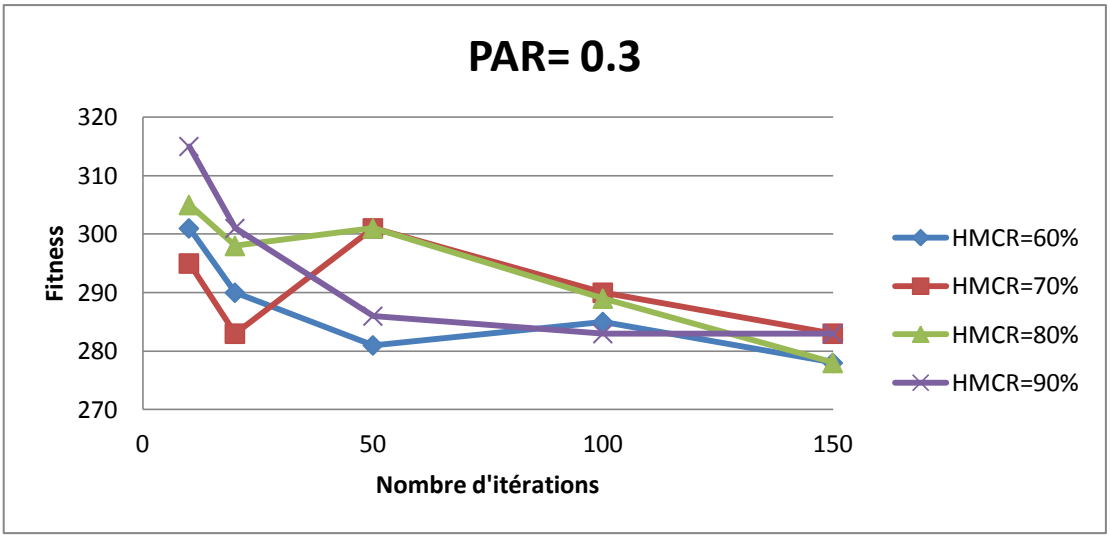


Figure 5. 6. Évolution de la fonction objectif pour différentes valeurs du HMCR et du PAR (Nc= 500)

En examinant les figures 5.5 et 5.6, nous pouvons remarquer que les valeurs des taux HMCR et PAR les plus adaptés à notre problème sont respectivement 90%, 10%

b.5. Influence du nombre de conteneurs

Pour étudier l'impact du nombre de conteneurs sur les résultats générés par l'approche proposée, différentes expérimentations sur différentes instances ont été effectuées pour HMS = 50 et $N_{itér}$ = 100.

Les résultats sont donnés par le tableau 5.5, ou F_i et F_f désignent les valeurs de la fonction fitness respectivement lors de la première et la dernière génération, respectivement.

Tableau 5. 5. Evolution de la fonction fitness avec le nombre de conteneurs

N_c	N_{Bloc}	n_1, n_2 , n_3	F_i	F_f	Taux d'amélioration (%)
64	3	3	23	12	91,67
125	6	3	55	34	61,76
343	7	4	219	165	32,73
729	12	4	501	405	23,70
1000	11	5	813	714	13,87

Les résultats montrent que pour les instances de taille moyenne et de taille grande, le taux d'amélioration de la fonction fitness est assez petit. Ceci est dû à la qualité de la solution générée lors de la première itération et la complexité du problème.

c. Etude comparative entre les deux approches de résolution (AG et RH) :

Dans cette section, nous allons comparer les résultats générés par les deux approches ; algorithme génétique et recherche harmonique et l'algorithme LIFO pour 4 cas différentes instances du problème :

Tableau 5. 6. Instances de test

Taille du problème	Nombre de bloc	Taille du Bloc
Nc= 50	1	$n_1=n_2=n_3=4$
Nc= 150	3	$n_1=n_2=n_3=4$
Nc= 250	5	$n_1=n_2=n_3=4$
Nc= 500	10	$n_1=n_2=n_3=4$
Nc= 700	12	$n_1=n_2=n_3=4$
Nc=900	16	$n_1=n_2=n_3=4$

La taille de la population initiale et le nombre d'itération sont fixé respectivement à N= 50 et $N_{itér}$=100.

Le principe de l'algorithme LIFO (Last In First Out) est de placer les conteneurs dans l'ordre de leur arrivée pour être stockés dans un bloc c'est-à-dire le dernier arrivé est le premier sorti. Ce principe est appliqué dans la plupart des terminaux à conteneurs du port, où le processus d'attribution de chaque conteneur à un bloc de stockage est basé sur l'expérience tout en respectant les contraintes relatives aux différents types de conteneurs.

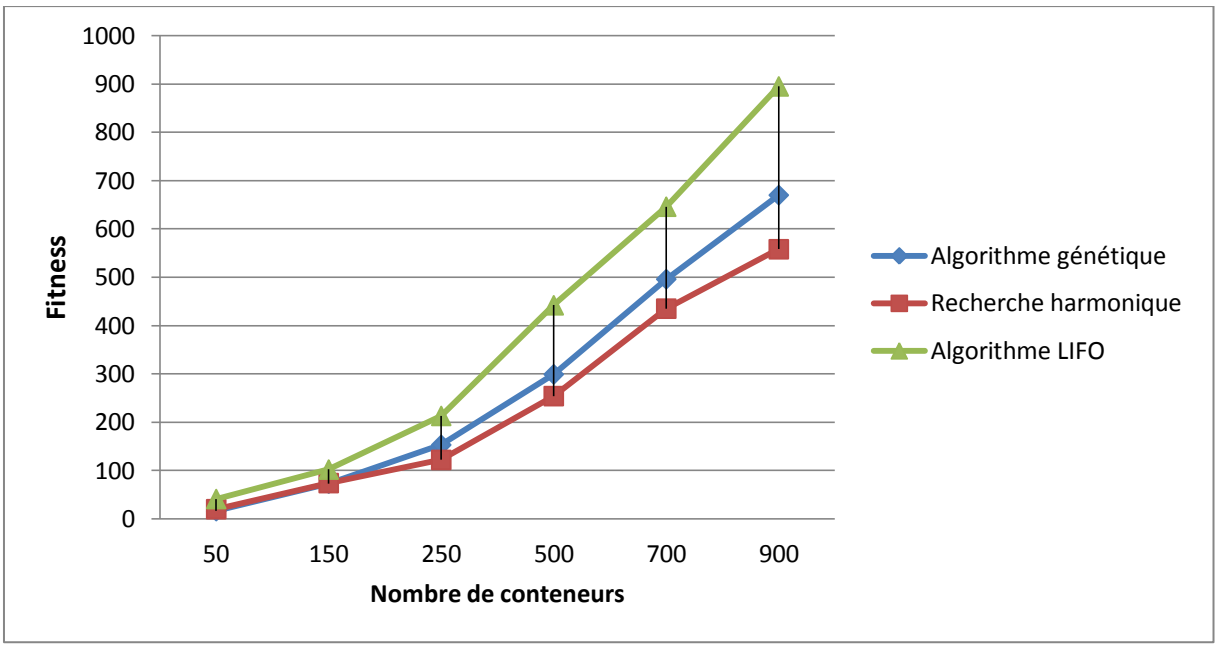


Figure 5. 7. Comparaison de la RH, l'AG et l'algorithme LIFO (Taille du problème)

Les données illustrées dans la figure 5.7 montrent que les résultats fournis par l'algorithme génétique et la recherche harmonique sont largement meilleurs en termes de fonction fitness que ceux fournis par l'algorithme LIFO surtout pour les problèmes de moyennes et grandes tailles. LIFO utilise la date d'arrivée comme critère d'organisation ce qui n'est pas toujours correcte.

Les deux approches AG et RH donnent des résultats presque identiques pour des problèmes de petites tailles.

Pour les problèmes de moyennes et grandes tailles, la recherche harmonique est la plus performante. Ainsi, elle génère des meilleurs résultats (qualité de fitness) en un temps plus rapide que l'approche génétique.

Ceci est dû principalement au processus de création d'une nouvelle solution. En effet, pour les AG, la création d'une nouvelle solution se base sur les opérateurs de sélection, croisement et de mutation. Ces opérateurs ne favorisent pas beaucoup la diversité des solutions et l'exploration totale de l'espace de recherche. Ainsi, des parties de la nouvelle solution seront similaires aux anciennes solutions (2 parents choisis). Néanmoins, pour la RH, la construction de la nouvelle solution se fait en choisissant les emplacements des conteneurs de diverses solutions de la mémoire harmonique avec une probabilité HMCR ou en générant une solution n'ayant aucune relation avec les anciennes solutions. En effet, les solutions générées par l'approche harmonique sont plus variées et exploitent mieux l'espace de recherche.

La limite de l'algorithme génétique reste toujours le temps de son exécution, sa convergence est lente vers la bonne solution ceci peut être expliqué par la création de toute une population pour chaque itération.

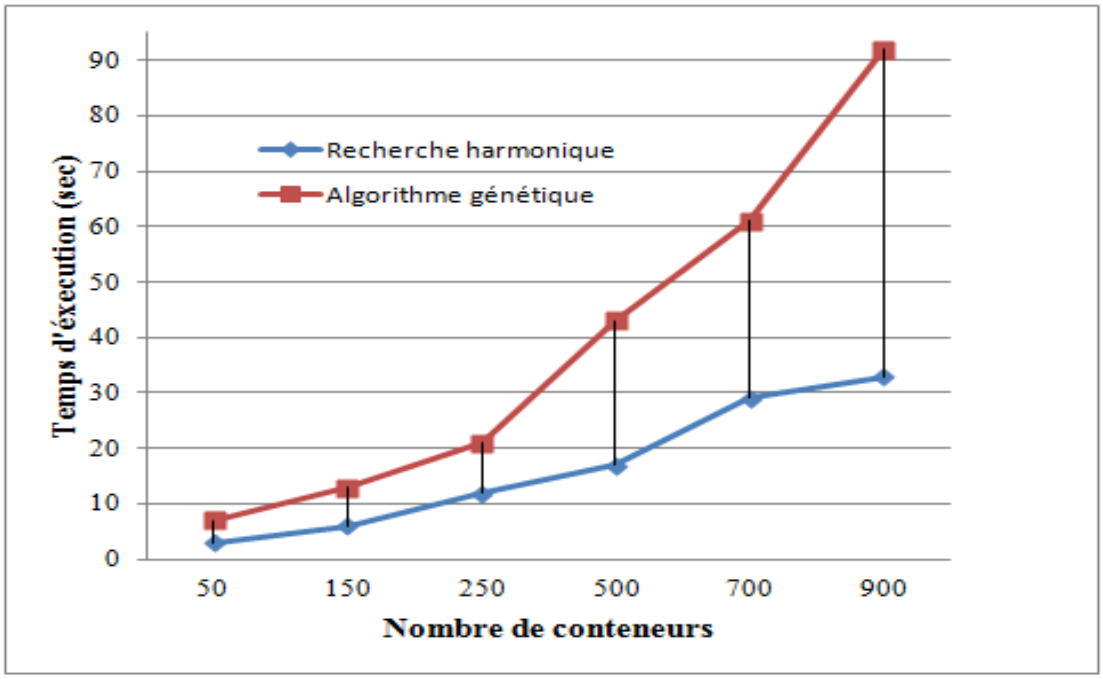


Figure 5. 8. Comparaison de la RH avec l'AG (Temps d'exécution)

2. Le PSC à plusieurs types de conteneurs

a. L'application de l'algorithme génétique à la résolution du PSC statique à plusieurs types de conteneur

L'algorithme génétique pour le PSC à différents types, décrit ci-dessus a été réalisé avec le même environnement de travail utilisé pour les approches décrites dans le chapitre précédent.

Lors de l'exécution de l'algorithme, il est supposé que :

- La taille de la population, *N*, varie entre 20 et 100.
- n_1, n_2 *et* n_3 sont identiques pour tous les blocks et sont définis par l'utilisateur.
- N_D : le nombre de types de conteneurs est défini par l'utilisateur
- *Nc(D)* : le nombre de conteneur pour chaque type D est à définir par l'utilisateur
- N_{stock_refrig}, N_{stock_reg} sont le nombre de blocks réfrigérés et le nombre de blocks standards, respectivement donnés par l'utilisateur.
- La date de départ de chaque conteneur est donnée par l'utilisateur.
- La probabilité de croisement est 80%
- La probabilité de mutation est de 20%.
- Le critère d'arrêt (nombre de générations, noté $N_{itér}$) est compris entre 25 et 150.

Plusieurs expérimentations ont été effectuées afin d'étudier l'influence de trois principaux paramètres, N_c, $N_{itér}$ et *N*.

a.1. Influence du critère d'arrêt

Afin d'étudier l'influence du critère d'arrêt sur les solutions générées par l'AG, le programme a été exécuté pour différentes valeurs de $N_{itér}$ tout en fixant N_D à 4, $N_c(1)=50$, $N_c(2)=50$, $N_c(6)=50$, N à 30, n1 = n2= n3=3 ; Les blocs de stockage sont initialement vides, Nstock_reg =4, Nstock_refrig = 2. Les résultats sont donnés par le tableau suivant :

Tableau 5. 7. Influence du critère d'arrêt (AG pour le PSC à plusieurs types)

$N_{itér}$	Fi	Ff	$T_{Execution}$(s)
25	66	23	71
50	68	16	80
75	61	12	85
100	72	10	91
150	60	10	95

Les résultats montrent que la qualité de la solution s'améliore lorsque $N_{itér}$ augmente. Mais l'exécution devient plus lente. Nous remarquons aussi qu'à partir de l'itération 100 l'algorithme a convergé vers la solution « 10 ».

a.2. Influence de la taille de la population

Pour évaluer l'influence de la taille de la population initiale sur le résultat généré, plusieurs simulations ont été établies pour différentes valeurs de *N*. Les différentes valeurs, données par le tableau 5.8, ont été enregistrées pour :

- $N_D = 4$ (à usage général, plate-forme, citerne et réfrigéré)
- n1= n2 = n3 = 3; Nstock_reg = 5, Nstock_refrig = 2.
- Nc(1) = 60, Nc(2) = 40, Nc(4) = 10, Nc(6) = 40.
- $N_{itér} = 50$

Tableau 5. 8. Influence de la taille de la population (AG pour le PSC à plusieurs types)

N	Fi	Ff	$T_{Execution}$ (s)
20	65	4	105
50	62	2	116
70	61	1	124
100	58	1	132
150	52	0	146

D'après les résultats du tableau, on remarque que la solution s'améliore lorsque la taille de la population augmente.

a.3. Influence du nombre de conteneurs

Dans cette analyse de sensibilité relative au paramètre $N_c(D)$, le nombre de conteneurs arrivant au port, l'algorithme a été exécuté pour différentes valeurs de N_D, le nombre de types de conteneurs.

Pour chaque simulation, les valeurs des meilleures solutions générées au niveau de la première et de la dernière itération (F_i et F_f, respectivement) ont été enregistrées. Pour cela, la taille de la population est fixée à 50, $N_{itér}$ à 100.

Les résultats sont donnés par le tableau 5.9 où le nombre de types de conteneurs ainsi que le nombre de conteneurs pour chaque type, F_i, F_f et le temps d'exécution sont enregistrés

Tableau 5. 9. Influence du nombre de conteneurs (AG pour le PSC à plusieurs types)

N_D	$N_c(D)$	Nombre du bloc	Taille du bloc	F_i	F_f	$T_{Execution}$ (s)
2	$N_c(1)=50, N_c(2)=50$	$N_{stock\_reg} = 6$	$n_1 = n_2 = n_3 = 3$	20	5	53
3	$N_c(1)=50$, $N_c(2)=50$, $N_c(3)=8$	$N_{stock\_reg} = 6$	$n_1 = n_2 = n_3 = 3$	62	11	62
4	$N_c(1)=50$, $N_c(2)=50$, $N_c(3)=20, N_c(4)=10$	$N_{stock\_reg} = 9$	$n_1 = n_2 = n_3 = 3$	61	5	68
5	$N_c(1)=50$, $N_c(2)=50$, $N_c(3)=20$, $N_c(4)=10$, $N_c(5)=50$	$N_{stock\_reg}=5$ $N_{stock\_refrig}= 1$	$n_1 = n_2 = n_3 = 4$	77	3	74
6	$N_c(1)=50$, $N_c(2)=50$, $N_c(3)=20$, $N_c(4)=10$, $N_c(5)=15$, $N_c(6)=50$	$N_{stock\_reg} = 5$ $N_{stock\_refrig} = 1$	$n_1 = n_2 = n_3 = 4$	83	15	124

Les résultats enregistrés sur les différentes instances montrent que l'AG a toujours trouvé une solution tolérable et optimale dans la majorité des cas en un temps raisonnable. On remarque aussi que l'algorithme devient moins rapide lorsque le nombre de types de conteneurs augmente et surtout quand le nombre de blocs disponible est limité. Ceci peut être expliqué par les contraintes de stockage à vérifier pour chaque type de conteneur ajouté. Ainsi, le stockage des conteneurs à toit ouvert provoque l'indisponibilité des emplacements au dessus et pour les conteneurs plate forme les places à coté de la partie ouverte seront aussi indisponibles.

b. L'application de l'algorithme de la recherche harmonique à la résolution du PSC statique à plusieurs types de conteneur

Les différentes expérimentations ont été effectuées en supposant que :

- La taille de la population, *N*, varie entre 20 et 100.
- n_1, n_2 *et* n_3 sont identiques pour tous les blocks et sont définis par l'utilisateur.
- N_D, le nombre de types de conteneurs est défini par l'utilisateur
- $N_c(D)$, le nombre de conteneurs pour chaque type D est à définir par l'utilisateur
- N_{stock_refrig}, N_{stock_reg}, sont le nombre de blocs réfrigérés et le nombre de blocs standards, respectivement donnés par l'utilisateur.
- La date de départ de chaque conteneur est donnée par l'utilisateur.
- Le taux de considération de la mémoire harmonique (HMCR) est de 90%
- Le taux d'ajustement (PAR) est de 10%.

- Le critère d'arrêt (nombre de générations, noté $N_{itér}$) est compris entre 25 et 150.

Les différentes expérimentations ont été réalisées en variant le nombre de types de conteneurs et la taille de la mémoire harmonique afin d'étudier leur influence sur le résultat final.

b.1. Influence du nombre d'itérations

Le nombre d'itérations est un paramètre très important pour toute méthode de résolution. Afin d'évaluer son influence sur la qualité des résultats de l'algorithme RH, nous avons réalisé des simulations en variant les valeurs de $N_{itér}$.

Nous avons choisi le cas suivant :

- $N_c(T)$ = 4 (général, vide, toit ouvert, réfrigéré) avec $Nc(1)$= 60, $Nc(2)$= 60 $Nc(3)$=10, $Nc(6)$=50.
- $HMS=50$.
- $n_1 = n_2 = n_3 = 4$.
- $N_{stock\_reg} = 3$, $N_{stock\_refrig} = 2$

Les résultats des simulations sont présentés par le tableau ci-dessous :

Tableau 5. 10. Influence du nombre d'itérations (RH pour le PSC à plusieurs types)

$N_{itér}$	Fi	Ff	$T_{Execution}$(s)
10	70	62	7
30	72	40	34
50	61	34	38
75	83	32	41
100	74	28	48
150	65	28	64

L'algorithme converge vers la solution optimale (F=28) à partir de 100 itérations. Un nombre d'itérations élevé nécessite plus de temps d'exécution.

b.2. Influence de la taille de la mémoire harmonique

Pour examiner l'importance de la taille de la mémoire harmonique, quelques simulations, avec différentes tailles de la HM, ont été faites pour :

- $N_c(T)$ = 5 (général, vide, toit ouvert, Citerne, réfrigéré) avec $Nc(1)$= 20, $Nc(2)$= 20 $Nc(3)$=15, $Nc(5)$= 10, $Nc(6)$=20.
- N_{iter} = 50
- $n_1 = n_2 = n_3 = 3$.

- $N_{stock\_reg} = 3$, $N_{stock\_refrig} = 3$

Les résultats sont présentés dans le tableau suivant :

Tableau 5. 11. Influence de la taille de la population (RH pour le PSC à plusieurs types)

HMS	F_i	F_f	$T_{Execution}$ (sec)
10	23	12	6
20	19	8	10
40	16	7	19
60	15	7	33
80	15	6	38
100	13	6	46

Les résultats montrent que l'augmentation de la mémoire harmonique améliore la qualité de la solution mais augmente le temps d'exécution.

b.3. Influence du nombre de types de conteneurs

Pour étudier l'influence du nombre de types de conteneurs sur la fonction fitness et le temps d'exécution de l'approche, la taille de la population a été fixée à 100, N_{iter} à 100

Les résultats sont donnés par le tableau 5.12, où le nombre de types de conteneurs ainsi que le nombre de conteneurs pour chaque type, F_i, F_f et le temps d'exécution sont présentés.

Tableau 5. 12. Influence du nombre de type de conteneurs (RH pour le PSC à plusieurs types)

N_D	$N_c(D)$	Nombre du bloc	Taille du bloc	F_i	F_f	$T_{Execution}$ (s)
2	$N_c(1)=100, N_c(2)=100$	$N_{stock\_reg}=8$	$n_1 = n_2 = n_3 = 3$	98	72	150
3	$N_c(1)=80$, $N_c(2)=80$, $N_c(3)=12$	$N_{stock\_reg}=8$	$n_1 = n_2 = n_3 = 3$	83	58	206
4	$N_c(1)=70$, $N_c(2)=70$, $N_c(3)=12, N_c(4)=12$	$N_{stock\_reg}=6$	$n_1 = n_2 = n_3 = 4$	104	46	101
5	$N_c(1)=80$, $N_c(2)=80$, $N_c(3)=12, N_c(6)=12$	$N_{stock\_reg}=6$	$n_1 = n_2 = n_3 = 4$	113	60	122
6	$N_c(1)=80$, $N_c(2)=70$, $N_c(3)=15$, $N_c(4)=15$, $N_c(6)=100$	$N_{stock\_reg}=6$ $N_{stock\_refrig}= 2$	$n_1 = n_2 = n_3 = 4$	115	61	136
7	$N_c(1)= 80$, $N_c(2) = 70$, $N_c(3) =15$, $N_c(4) = 15$, $N_c(5)=10$, $N_c(6)=100$	$N_{stock\_reg}=6$ $N_{stock\_refrig}= 2$	$n_1 = n_2 = n_3 = 4$	128	70	149

D'après ce tableau, on remarque que l'approche harmonique génère toujours une solution tolérable. D'autre part, le temps d'exécution augmente avec le nombre de types considérés. Ceci s'explique par le fait que la complexité du problème traité est directement reliée aux contraintes de chacun des types de conteneurs.

c. Etude comparative entre les deux approches de résolution :

Afin d'évaluer les résultats générés par l'AG, la RH, une étude comparative de ces deux approches et de l'algorithme LIFO a été réalisée.

Cette étude concerne une zone de stockage composé de 5 blocs conçus pour les conteneurs réfrigérés et 5 blocs généraux avec $n_1=n_2=n_3=5$.

Initialement, dans l'espace de stockage, il y a des conteneurs répartis comme suit :

- 20 conteneurs de type général.
- 15 conteneurs à toit ouvert.
- 20 conteneurs de type réfrigéré.
- 15 citernes.
- 20 conteneurs vides.

Une description détaillée de la répartition des conteneurs dans les blocs de stockage est donnée dans l'annexe 1 (état initial)

Pour vérifier la performance de chaque approche, nous avons conçu 5 instances décrites dans le tableau ci-dessous. Pour chaque test, nous présentons le nombre de contenteurs de chaque type que nous devons placer dans l'espace de stockage.

Tableau 5. 13. Instances du PSC statique à plusieurs types

Type de conteneur	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5
Général	20	50	70	100	120
Toit ouvert	0	10	20	20	25
réfrigéré	20	30	50	100	100
Citernes	15	20	50	70	75
Vides	20	30	50	100	100
Plate-forme	10	10	20	20	25

Pour chaque instance, les 2 approches AG, RH ont été exécutés 10 fois et les moyennes des fonctions fitness générées des temps d'exécution sont données. Les expérimentations ont été effectuées avec une population de taille 50, N_{iter} = 100.

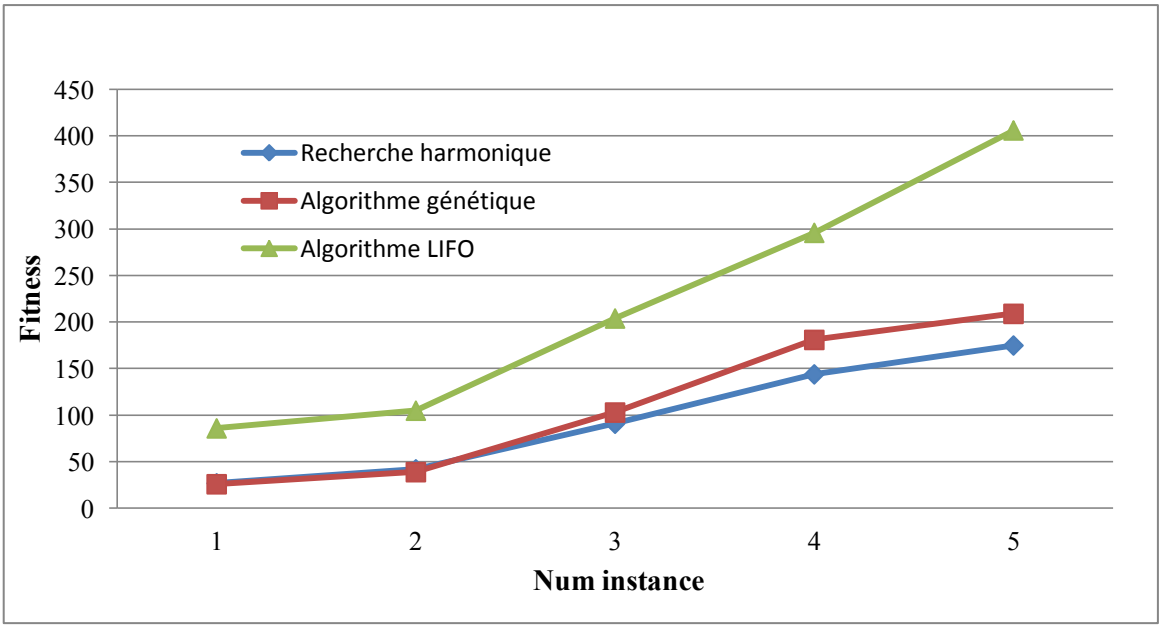


Figure 5. 9. Comparaison entre AG, RH et LIFO (Fitness)

Les données illustrées dans la figure 5.9 montrent que les résultats fournis par l'algorithme génétique et la recherche harmonique sont largement meilleurs en termes de fonction fitness que ceux fournis par l'algorithme LIFO.

Les résultats des deux approches RH et AG sont similaires pour les deux premiers tests ou le nombre de conteneurs à stocker est moyen et les conteneurs ayant des conditions de stockages assez difficle ne sont pas nombreux (toit ouvert, plate forme, citerne).

La recherche harmonique a prouvé son éfficacité et sa supériorité par rapport à l'algorithme génétique pour les 3 dernières instances. Ces tests comprend un nombre de conteneurs élevé de divers types ce qui rend le problème de plus en plus compliqué. Ainsi, le stockage d'un conteneur de type toit ouvert provoque l'indisponibilité de tous les emplacements au dessus. La même situation se présente pour les conteneurs plate forme avec une indisponibilité des positions qui se trouvent sur tout l'espace du coté ouvert. Aussi, un conteneur citerne ne peut se placer que par terre ou sur un autre de même type.

La figure 5.10 présente le temps d'exécution des 2 approches pour chaque instance. L'algorithme génétique souffre toujours de sa lenteur de convergence.

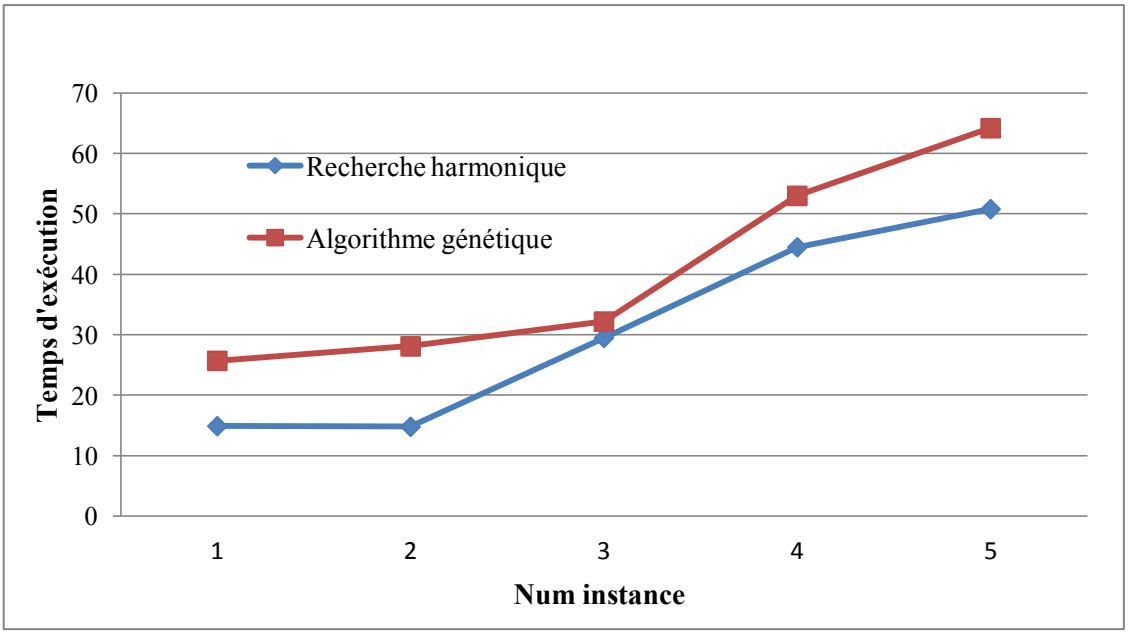


Figure 5. 10. Comparaison entre AG, RH et LIFO (Temps d'exécution)

II. Le PSC dynamique : Simulation et résultats

Dans cette partie, nous allons présenter notre étude pour la résolution du PSC dynamique où l'on doit satisfaire les changements des informations concernant des conteneurs qui sont déjà en stock. Nous traitons dans ce travail les modifications des dates de départ des conteneurs ainsi que la séquence de déchargement.

Comme nous l'avons décrit dans le chapitre précédent, nous avons développé deux méthodes différentes pour gérer ses changements. La première méthode utilise une procédure de réarrangement (remarshalling) pour modifier l'organisation des conteneurs de manière qu'il s'adapte mieux aux nouvelles données et un algorithme de recherche harmonique est appliqué par la suite pour déterminer les emplacements des nouveaux conteneurs. La deuxième méthode consiste à retirer les conteneurs ayant des dates de départ modifié ainsi que ceux qui sont au dessus des blocs de stockage et les ajouter à la liste des nouveaux conteneurs. L'algorithme de la recherche harmonique est appliqué ainsi pour déterminer des emplacements à cet ensemble des conteneurs.

L'objectif des deux approches est la minimisation du nombre de mouvements de remaniements qui se produisent pour modifier ou améliorer l'organisation des conteneurs initiales ainsi que le nombre de mouvements parasites au moment de déchargement de tous les conteneurs existant dans la zone de stockage.

La fonction objectif est décrite par l'équation suivante :

$$\text{Min}\left(\sum_{cm=1}^{N_{cm}} m_{cm} + \sum_{D=1}^{N_D}\sum_{i=}^{Nc(D)}\sum_{b=1}^{N_{Bloc}} m_{i,b}\, C_{i,D}(x, y, z, b)\right) \qquad [5.1]$$

avec:

- m_{cm} : le nombre de mouvements cumulés à la $n^{ème}$ période lorsque des modifications sont apportées à un conteneur cm.
- $m_{i,b}$: le nombre minimum de conteneurs manipulés pour retirer le conteneur i dans le bloc b

Pour vérifier la performance des deux solutions proposées pour la résolution du PSC dynamique, nous présentons une étude comparative des deux méthodes. En effet, chaque approche est appliquée pour résoudre un ensemble d'instances de PSC dynamique.

Pour chaque instance de test, nous présentons les conteneurs qui sont déjà dans les blocs et ayant des dates de départ modifiés (avancée, retardée) ainsi que leurs anciennes et nouvelles dates.

La liste des nouveaux conteneurs nécessitant leur stockage est la même pour chaque instance. Elle est composé de :

- 30 conteneurs de type général.
- 20 conteneurs à toit ouvert.
- 20 citernes.
- 30 conteneurs vides.

Les instances du problème sont présentées dans le tableau ci-dessous.

Tableau 5. 14. Instances du PSC dynamique

Instance 1			Instance 2			Instance 3			Instance 4		
N°cont	date_A	date_N	N°cont	date_A	date_N	N°cont	date_A	date_N	N°cont	date_A	date_N
219	19	40	219	19	40	219	19	40	219	19	40
623	23	90	623	23	90	623	23	90	623	23	90
648	48	80	648	48	80	648	48	80	648	48	80
226	26	70	226	26	70	226	26	70	226	26	70
641	41	56	641	41	56	641	41	56	641	41	56
			719	19	42	719	19	42	719	19	42
			724	24	45	724	24	45	724	24	45
			620	20	42	620	20	42	620	20	42
			732	32	50	732	32	50	732	32	50
			62	1	80	62	1	80	62	1	80
						731	31	51	731	31	51
						60	60	85	60	60	85
						29	29	71	29	29	71
						18	18	82	18	18	82
						3	3	50	3	3	50
									36	36	45
									152	52	70

			126	26	75
			33	33	53
			148	48	75

Cette étude concerne une zone de stockage composé de 3 blocs conçus pour les conteneurs réfrigérés et 7 blocs généraux avec $n_1=n_2=n_3=5$.

Initialement, dans l'espace de stockage, il y a des conteneurs répartis comme suit :

- 70 conteneurs de type général.
- 30 conteneurs à toit ouvert.
- 30 conteneurs de type plate-forme
- 70 conteneurs de type réfrigéré.
- 50 citernes.
- 70 conteneurs vides.

Une description détaillée de la répartition des conteneurs dans les blocs de stockage est donnée dans l'annexe 2 (état initial).

Les résultats de simulation présentés dans la figure 5.11 montrent une légére supériorité de la 1ère approche par rapport à la seconde. En effet, l'utilisation de la méthode de réarrangement des blocs de stockage suivi d'une application de l'algorithme de la recherche harmonique pour l'affectation des nouveaux conteneurs génére de meilleures solutions.

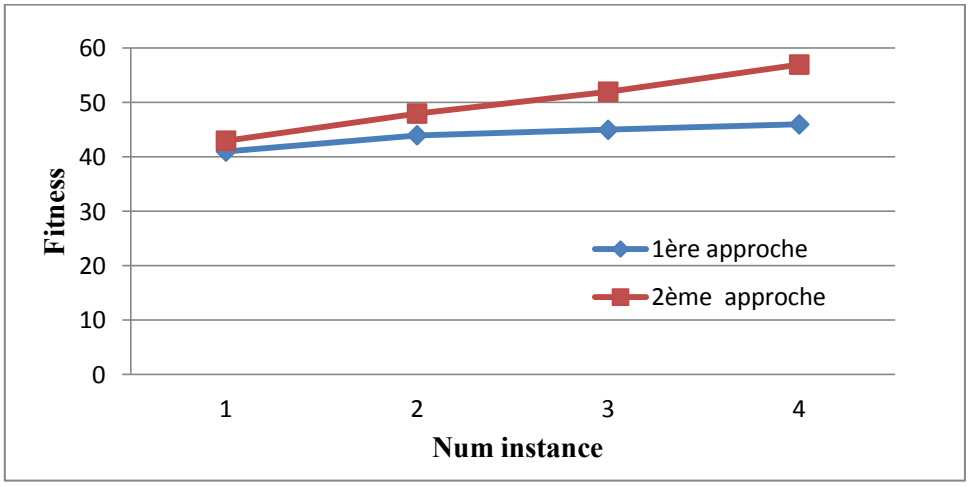


Figure 5. 11. Comparaison entre les 2 approches du PSC dynamqiue (Fitness)

Une approche efficace ne doit pas uniquement générer de bons résultats mais elle doit aussi les déterminer en un temps de calcul assez rapide.

Le temps d'exécution mis par les 2 approches est toléarble. Néaumoins, la 2ème méthode nécessite moins du temps pour générer sa solution. Ceci est dû au faite que la 1ére approche

nécessite un temps supplémentaire pour appliquer la méthode de réarrangement et générer le nouveau plan de stockage initial avant d'utiliser l'algorithme de la RH.

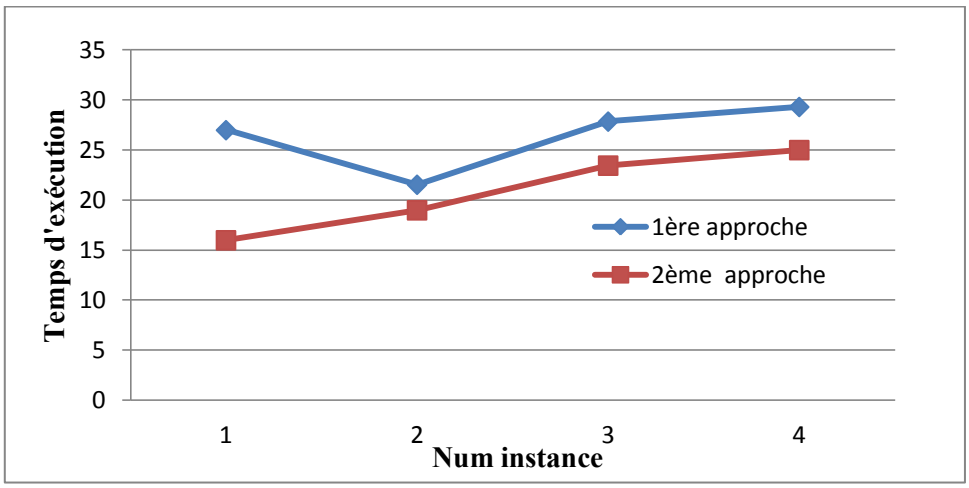


Figure 5. 12. Comparaison entre les 2 approches du PSC dynamqiue (temps d'exécution)

Conclusion

Dans ce chapitre, nous avons présenté les résultats trouvés pour les différents problèmes traités : le PSC statique à un seul et plusieurs types de conteneurs et le PSC dynamique à plusieurs types de conteneurs.

Nous avons utilisé deux métaheuristiques, l'algorithme génétique et la recherche harmonique comme outil de résolution pour le cas statique ou la RH a prouvé sa supériorité. En effet, nous avons gardé cette métaheuristique pour la résolution du PSC dynamique.

Pour traiter les incertitudes des dates de départ des conteneurs, nous avons proposé deux approches. La première se base sur l'utilisation d'une méthode de réorganisation des blocs de stockage. La seconde adopte l'algorithme de la recherche harmonique.

Pour tester les différentes approches nous avons opté pour la construction d'un ensemble de benchmarks qui représentent des cas permettant de juger l'efficacité et la réactivité des approches de résolution.

Pour vérifier la performance et l'efficacité de chaque méthode de résolution, nous avons comparé leurs résultats.

Conclusion générale

Toute ouvre scientifique « achevée » n'a d'autre sens
que celui de faire naître de nouvelles « questions»,
elle demande donc à être «dépassée » et à vieillir.
Dans les sciences, non seulement notre destin,
mais encore notre but à tous est de nous voir
un jour dépassés.
[Max Webber]

Conclusion générale

Dans un terminal à conteneurs, l'opération de chargement et de déchargement des conteneurs est une tâche très critique vu qu'elle a une influence directe sur la productivité des opérations de déchargement des conteneurs destinés à l'import et à l'export.

Le stockage inapproprié des conteneurs peut détériorer la productivité du terminal puisque l'opération de récupération de ces conteneurs au moment de leurs départs peut devenir inefficace et donc provoquer des retards. Parfois, les conteneurs peuvent être temporairement relocalisés ou remaniée afin de récupérer un conteneur cible empilés en dessous. À d'autres moments, les grues d'empilage peuvent avoir à parcourir une longue distance pour récupérer les conteneurs qui doivent être chargés de façon consécutive. Ces opérations supplémentaires de manutention et la distance de transfert étendue de grues d'empilage sont la principale source de retard de l'opération de chargement.

En effet, un bon plan d'arrangement de ces conteneurs aux différents emplacements disponibles dans les blocs de stockage minimise le temps de leur déchargement et diminue par la suite le retard des différents moyens de transport (navire, voies ferroviaires et routières).

Le travail de recherche que nous présentons traite le problème de stockage des conteneurs, (PSC) dans le cas statique (date d'arrivée et de départ des conteneurs fixes) et dynamique (les dates d'arrivée et de départ sont incertaines et aléatoires et parfois inattendues). Ce problème est NP-difficile et NP-complet.

Nous avons choisi les deux métaheuristiques, les algorithmes génétiques et la recherche harmonique pour le résoudre. Notre objectif était toujours la minimisation des nombres de conteneurs remaniés lors du processus de déchargement (départ) des conteneurs du port.

Le PSC est souvent étudié en considérant un seul type de conteneur ce qui n'est pas le cas dans le monde réel. Il existe beaucoup de type de conteneurs manipulés dans les opérations portuaires à savoir les citernes, les conteneurs toit ouvert, plate-forme, réfrigéré,…
La littérature du PSC à plusieurs types est rare, vu que chaque type possède un ensemble de contraintes de stockage. Ceci complique la formulation du problème et rend sa résolution beaucoup plus difficile.

Avant d'aborder la résolution de ce problème, nous avons présenté un état de l'art sur les terminaux à conteneurs, les opérations portuaires et les différents problèmes rencontrés dans un port maritime. Nous avons détaillé par la suite le problème étudié dans cette thèse, « le problème de stockage des conteneurs : PSC ». Nous avons ainsi présenté un état de l'art de ce problème ainsi que les différentes méthodes de résolution proposées.

Ensuite, nous avons décrit les deux métaheuristiques utilisées pour la résolution du PSC statique ; l'algorithme génétique et la recherche harmonique.

L'objectif de cette thèse est d'étudier trois variantes du PSC (problème statique à un seul type de conteneur, problème statique à plusieurs types de conteneurs et problème dynamique à plusieurs types de conteneurs) et de proposer des solutions pour chacun d'entre elles.

Dans le troisième chapitre, consacré à l'aspect statique du problème, nous avons présenté les détails des quatre approches proposées :

- Algorithme génétique pour la résolution du PSC statique à un seul type de conteneurs.
- Algorithme de la recherche harmonique pour la résolution du PSC statique à un seul type de conteneurs.
- Algorithme génétique pour la résolution du PSC statique à un plusieurs types de conteneurs.
- Algorithme de la recherche harmonique pour la résolution du PSC statique à un plusieurs types de conteneurs.

Dans le quatrième chapitre, nous avons développé deux méthodes pour faire façe aux incertitudes des dates de départ des conteneurs initialement dans l'espace de stockage. La première approche utilise une méthode de réarrangement. La seconde consiste à ajouter l'ensemble des conteneurs dont la date a été modifié et ceux qui leurs sont au dessus aux nouveaux conteneurs arrivant. Par la suite, après la modification de l'état initial de la zone de stockage, le problème est résolu comme un PSC statique en utilisant l'algorithme de la recherche harmonique.

Dans le cinquième et dernier chapitre, nous avons présenté les outils informatiques développés pour la résolution du PSC statique et dynamique. Nous avons aussi exposé les résultats des simulations effectuées sur des tests. Les résultats indiquent la performance des différentes approches proposées par rapport aux outils utilisés par les planificateurs dans les ports (algorithme LIFO). La recherche harmonique a montré sa performance surtout pour les problèmes statiques complexes. Pour cela la RH a été adoptée pour la résolution des problèmes dynamiques après réorganisation de l'état initial de l'espace de stockage. Les résultats génères par la méthode de réarrangement sont meilleurs pour les différentes instances du problème.

Enfin, comme perspectives de nos travaux de recherche, nous envisageons de :

- Développer d'autres méthodes de réarrangement (Remarshalling) pour le PSC dynamique.
- Ajouter d'autres contraintes relatives au processus de stockage à savoir le poids des conteneurs et leurs destinations (classification des conteneurs selon les navires de destination).
- Tester nos approches sur un cas pratique (port de Radès à Tunis).
- Adopter les approches proposées pour la résolution du PSC statique et dynamique au problème d'arrimage de conteneurs dans un navire pour déterminer un plan de chargement.
- Intégrer d'autres algorithmes, tels que la recherche Tabou ou les colonies de fourmis pour la résolution du PSC statique et dynamique.
- Généralisation des approches développées pour la résolution des problèmes du Bin packing statiques et dynamiques à trois dimensions.

Annexes

Annexe 1

Etat initial :

Bloc 1
204 215 219 202 209 210 201 206 218 205 217 214 207 211 212 203 213 216 220 208 415 13 16 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0
Bloc 2
1411 410 414 19 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1401 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1415 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1414 999 999 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1410 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 3
1402 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 703 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
707 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 705 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 704
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 4
402 408 4 20 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 5

1215 1206 1204 1203 1212 1202 1216 1211 1209 1201 1205 1220 1217 1213 1214 1219 1218 1208
1207 1210 404 5 18 10 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 999 0 0 0 0
Bloc 6
1408 405 403 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1406 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1413 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1412 999 999 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1409 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 7
406 15 17 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 8
401 14 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 9
409 412 407 413 9 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 10
411 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Instance 1

Algorithme de la recherche harmonique

Bloc 1
204 215 219 202 209 210 201 206 218 205 217 214 207 211 212 203 213 216 220 208 415 13 16 36
620 119 34 40 611 37 35 33 30 32 27 619 26 29 118 31 617 21 38 618 120 999 28 615 614 616
22 25 406 999 999 404 999 999 999 999 104 24 114 613 410 39 117 23 603 409 999 113 403 999 999
408 999 999 999 999 999 999 999 999 999 101 407 999 999 999 405 999 999 999 999 999 402 999
999 999 999 999 999 999 999 999 999 999 999 999 401 999 999 999 999 999 999 999 999 999 999
999 999 999 999
Bloc 2
1411 410 414 19 1 713 712 715 714 111 105 116 102 612 610 607 609 112 711 606 709 704 107 608
110 1401 999 999 605 115 710 708 707 706 109 604 108 106 103 602 601 0 0 705 0 703 702 0 0 0
1415 999 999 0 0 701 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1414 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1410 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 3
1402 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1403 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1407 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1405 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1404 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 4
402 408 4 20 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 5
1215 1206 1204 1203 1212 1202 1216 1211 1209 1201 1205 1220 1217 1213 1214 1219 1218 1208
1207 1210 404 5 18 10 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 999 0 0 0 0
Bloc 6
1408 405 403 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1406 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1413 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1412 999 999 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1409 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 7
406 15 17 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 8
401 14 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 9
409 412 407 413 9 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 10
411 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Algorithme génétique
Bloc 1
204 215 219 202 209 210 201 206 218 205 217 214 207 211 212 203 213 216 220 208 415 13 16 28
410 614 110 609 27 408 406 999 999 999 999 605 602 0 0 0 0 0 0 0 0 999 0 0 0 999 0 0 0 0 999 999
999 999 999 999 0 0 0 0 0 0 0 0 0 0 999 0 0 0 999 0 0 0 0 999 999 999 999 999 999 0 0 0 0 0 0 0 0 0
0 999 0 0 0 999 0 0 0 0 999 999 999 999 999 999 0 0 0 0 0 0 0 0 0 0 999 0 0 0 999
Bloc 2
1411 410 414 19 1 617 601 715 115 603 30 0 0 101 0 113 0 0 0 0 0 613 0 0 0 1401 999 999 0 409 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1415 999 999 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1414 999
999 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1410 999 999 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
Bloc 3
1402 8 37 0 0 0 702 0 0 404 608 0 616 618 405 0 0 0 0 0 0 23 0 612 604 1403 0 0 0 0 0 0 0 0 999 0 0
0 0 999 0 0 0 0 0 0 0 0 0 0 1407 0 0 0 0 0 0 0 0 999 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 1405 0 0 0 0 0 0 0
0 999 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 1404 0 0 0 0 0 0 0 0 999 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0
Bloc 4
402 408 4 20 7 615 710 0 712 620 108 33 0 0 607 0 0 0 611 0 0 0 606 0 112 999 999 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 5
1215 1206 1204 1203 1212 1202 1216 1211 1209 1201 1205 1220 1217 1213 1214 1219 1218 1208
1207 1210 404 5 18 10 6 0 0 0 21 0 0 0 103 0 0 0 0 619 0 0 114 0 0 0 610 999 0 0 116 118
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0
Bloc 6
1408 405 403 703 0 26 0 0 0 0 25 0 0 706 31 403 999 999 999 999 0 36 0 0 0 1406 999 999 0 0 0 0 0
0 0 0 0 0 0 0 999 999 999 999 999 0 0 0 0 0 1413 999 999 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999
999 0 0 0 0 0 1412 999 999 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 999 0 0 0 0 0 1409 999 999 0 0 0
0 0 0 0 0 0 0 0 0 999 999 999 999 999 0 0 0 0 0
Bloc 7

406 15 17 2 0 111 713 0 119 117 35 707 0 701 24 0 0 0 39 0 106 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 8
401 14 12 0 105 107 40 0 0 705 0 0 34 0 0 0 0 109 0 0 0 0 0 0 0 999 0 32 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 9
409 412 407 413 9 3 0 0 0 711 22 709 0 0 0 0 0 0 0 0 0 0 120 0 29 999 999 999 999 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 104 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 10
411 11 0 0 38 708 0 102 401 999 407 999 999 999 999 0 0 0 402 999 714 0 0 0 0 999 0 0 0 0 704 0 0
999 999 999 999 999 999 999 0 0 0 999 999 0 0 0 0 0 999 0 0 0 0 0 0 0 999 999 999 999 999 999 999
0 0 0 999 999 0 0 0 0 0 999 0 0 0 0 0 0 0 999 999 999 999 999 999 999 0 0 0 999 999 0 0 0 0 0 999 0
0 0 0 0 0 0 999 999 999 999 999 999 999 0 0 0 999 999 0 0 0 0 0

Instance 2

Algorithme de la recherche harmonique

Bloc 1
204 215 219 202 209 210 201 206 218 205 217 214 207 211 212 203 213 216 220 208 415 13 16 50
143 150 148 145 147 149 48 43 47 49 45 42 146 102 210 141 142 46 40 140 144 999 36 138 38 133
24 34 37 28 139 627 44 31 39 29 134 129 41 999 126 33 630 26 132 209 999 136 130 629 135 106
208 32 117 35 408 999 999 999 999 203 113 23 999 407 21 410 999 999 999 999 30 405 999 999
207 999 409 999 999 999 999 999 999 999 999 27 25 999 999 406 999 999 999 999 999 404 999 999
999
Bloc 2
1411 410 414 19 1 628 137 128 131 625 125 626 622 624 127 118 123 119 22 623 621 120 121 620
124 1401 999 999 122 619 610 116 716 114 717 711 612 618 602 606 108 403 999 999 999 613 402
999 999 999 1415 999 999 107 109 115 112 720 205 719 206 608 101 204 617 702 999 999 999 999
201 999 999 999 999 1414 999 999 616 105 401 999 999 999 999 999 111 615 999 202 718 999 999
999 999 999 999 999 999 999 1410 999 999 614 110 999 999 999 999 999 999 104 611 999 999 715
999 999 999 999 999 999 999 999 999
Bloc 3
1402 8 713 714 609 709 707 712 607 605 708 710 603 604 706 103 701 704 705 601 703 0 0 0 0
1403 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1407 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1405 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1404 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 4
402 408 4 20 7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 5
1215 1206 1204 1203 1212 1202 1216 1211 1209 1201 1205 1220 1217 1213 1214 1219 1218 1208
1207 1210 404 5 18 10 6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 999 0 0 0 0
Bloc 6
1408 405 403 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1406 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1413 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1412 999 999 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1409 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 7
406 15 17 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Bloc 8
401 14 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 9
409 412 407 413 9 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 10
411 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

<u>Algorithme génétique</u>

Bloc 1
204 215 219 202 209 210 201 206 218 205 217 214 207 211 212 203 213 216 220 208 415 13 16 708
702 42 103 201 0 205 0 0 0 118 410 0 0 137 629 0 0 0 35 0 615 999 613 0 0 0 0 0 999 0 999 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 999 0 999 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 999 0
999 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0
Bloc 2
1411 410 414 19 1 626 621 0 123 0 0 710 619 36 0 0 707 130 101 0 720 102 604 620 0 1401 999 999
602 406 0 136 0 0 0 0 0 0 0 0 0 0 0 0 0 0 31 128 0 1415 999 999 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1414 999 999 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1410 999 999 0 999 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc3
1402 8 0 141 28 133 25 0 49 0 0 0 0 26 131 625 617 121 630 0 48 0 603 0 107 1403 0 0 402 999 0 0
0 0 0 0 0 0 0 0 0 0 404 999 999 0 0 0 0 405 1407 0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 0 0
0 0 999 1405 0 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 0 0 0 0 999 1404 0 0 999 999 0 0 0 0 0
0 0 0 0 0 0 0 999 999 999 0 0 0 0 999
Bloc 4
402 408 4 20 7 0 610 0 606 112 0 0 628 607 0 709 0 717 624 127 612 142 623 627 401 999 999 407
999 999 0 0 0 0 21 0 0 203 209 0 0 0 0 622 0 0 0 0 605 999 999 999 999 999 999 0 0 0 0 0 0 0 999
999 0 0 0 0 0 0 0 0 0 0 999 999 999 999 999 999 0 0 0 0 0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 999 999
999 999 999 999 0 0 0 0 0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 999
Bloc 5
1215 1206 1204 1203 1212 1202 1216 1211 1209 1201 1205 1220 1217 1213 1214 1219 1218 1208
1207 1210 404 5 18 10 6 614 37 0 0 106 616 0 111 0 0 0 145 0 0 0 601 45 611 0 0 999 608 618 609 0
135 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0
Bloc 6
1408 405 403 0 0 0 0 703 706 129 0 0 27 716 403 150 105 138 108 0 143 40 0 713 712 1406 999 999
0 0 0 0 0 0 0 0 0 0 0 999 140 0 0 0 0 0 0 0 0 0 1413 999 999 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0
0 1412 999 999 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 1409 999 999 0 0 0 0 0 0 0 0 0 0 0 999 0
0 0 0 0 0 0 0 0 0
Bloc 7
406 15 17 2 0 117 0 0 0 115 38 23 120 113 206 146 0 0 0 204 0 0 0 0 0 999 0 0 0 0 22 0 0 0 409 0
124 0 0 999 46 0 0 0 999 0 0 0 0 0 999 0 0 0 0 0 0 0 0 999 0 0 0 0 999 0 0 0 0 999 0 0 0 0 0 999 0 0 0
0 0 0 0 0 999 0 0 0 0 999 0 0 0 0 999 0 0 0 0 0 999 0 0 0 0 0 0 0 0 999 0 0 0 0 999 0 0 0 0 999 0 0 0 0
0
Bloc 8
401 14 12 718 50 0 0 0 0 714 109 104 207 0 0 33 0 32 34 41 147 0 144 134 711 999 30 0 0 29 0 0 0 0
0 43 0 999 0 0 0 0 0 0 0 0 0 0 132 0 999 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0
0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0

Bloc 9
409 412 407 413 9 3 0 0 0 715 0 0 114 210 0 0 119 0 122 116 0 110 0 149 0 999 999 999 999 139 24 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 44 0 202 0 999 999 999 999 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 999 0 999 999 999 999 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 999 0 999 999 999 999 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 999 0
Bloc 10
411 11 705 0 0 719 0 0 0 0 0 47 0 0 408 0 125 208 0 148 0 126 701 704 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 999 0 39 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 999 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 999 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 999 0 0 0 0 0 0 0

Instance 3

Algorithme génétique

Bloc 1
204 215 219 202 209 210 201 206 218 205 217 214 207 211 212 203 213 216 220 208 415 13 16 210 627 144 70 621 601 0 608 119 0 165 0 0 61 612 25 624 0 0 603 111 0 999 0 0 999 614 0 0 605 55 0 168 402 999 999 999 0 0 52 0 0 0 0 0 0 22 0 999 0 0 999 412 0 0 130 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 999 999 0 0 405 999 999 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 999 0 0 999 999
Bloc 2
1411 410 414 19 1 29 625 0 45 714 616 709 602 636 0 120 642 0 610 0 0 609 644 623 637 1401 999 999 619 634 0 115 0 0 0 0 0 0 118 0 0 0 413 999 999 999 0 607 0 126 154 1415 999 999 0 622 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 401 999 999 999 1414 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 999 999 999 999 1410 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 999 999 999 999
Bloc3
1402 8 0 420 999 615 0 163 205 617 626 640 0 0 606 0 631 613 648 133 706 161 732 725 628 1403 0 0 999 999 0 0 53 999 206 618 216 0 0 406 0 148 213 142 43 0 107 0 0 166 1407 0 0 999 999 0 0 21 999 999 137 999 0 0 999 0 26 999 122 0 0 0 0 0 57 1405 0 0 999 999 0 0 0 999 999 0 999 0 0 999 0 0 999 0 0 0 0 0 0 40 1404 0 0 999 999 0 0 0 999 999 0 999 0 0 999 0 0 999 0 0 0 0 0 0 0
Bloc 4
402 408 4 20 7 0 135 746 36 113 711 739 211 633 630 0 745 727 638 127 632 635 62 645 159 999 999 103 146 151 0 417 999 999 999 0 0 999 31 415 0 0 713 0 30 0 125 49 150 102 999 999 0 208 0 0 999 999 999 999 0 0 999 0 999 0 0 0 0 24 0 0 0 418 999 999 999 0 999 0 0 999 999 999 999 0 0 999 0 999 0 0 0 0 0 0 0 0 999 999 999 999 0 999 0 0 999 999 999 999 0 0 999 0 999 0 0 0 0 0 0 0 0 999 999
Bloc 5
1215 1206 1204 1203 1212 1202 1216 1211 1209 1201 1205 1220 1217 1213 1214 1219 1218 1208 1207 1210 404 5 18 10 6 47 121 0 0 209 641 647 110 0 0 629 646 643 0 0 170 649 0 0 164 999 0 639 50 650 0 0 0 0 999 124 212 0 0 0 620 59 611 0 0 131 0 0 0 0 999 0 169 0 403 0 0 0 0 999 220 999 0 0 0 604 34 410 999 999 408 999 999 999 999 999 0 69 0 999 0 0 0 0 999 999 999 0 0 0 0 0 999 999 999 999 999 999 999 999 999 999 0 0 0 999
Bloc 6
1408 405 403 750 716 32 158 0 104 0 0 0 141 708 0 132 705 733 721 56 0 701 219 160 735 1406 999 999 0 0 0 0 0 0 0 0 0 38 0 0 66 0 722 0 0 0 0 999 0 0 1413 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 39 0 0 0 0 0 0 0 0 999 0 0 1412 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 37 0 0 0 0 0 0 999 0 0 1409 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0
Bloc 7
406 15 17 2 109 138 729 0 162 0 0 147 702 0 0 704 0 749 157 0 23 0 117 58 0 999 155 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 744 35 0 0 0 0 0 0 999 105 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 8
401 14 12 0 136 0 145 215 217 409 67 726 0 737 419 748 0 0 747 740 108 738 44 128 143 999 0 112 0 204 0 116 999 999 999 0 703 0 0 999 0 0 0 0 730 207 0 0 114 214 999 0 0 0 999 0 0 999 999 999 0 0 0 0 999 0 0 0 0 723 999 0 0 106 999 999 0 0 0 999 0 0 999 999 999 0 0 0 0 999 0 0 0 0 717 999 0 0 28 999 999 0 0 0 999 0 0 999 999 999 0 0 0 0 999 0 0 0 0 0 999 0 0 0 999
Bloc 9

409 412 407 413 9 3 715 743 0 167 0 736 27 734 123 742 134 0 719 404 152 203 720 54 724 999 999 999 999 999 68 153 0 0 0 64 0 0 0 731 407 0 46 0 0 999 414 999 999 999 999 999 999 999 999 0 0 0 0 0 60 0 0 0 0 999 0 0 0 0 999 999 999 999 999 999 999 999 999 999 0 0 0 0 0 51 0 0 0 0 999 0 0 0 0 999 999 999 999 999 999 999 999 999 999 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 999 999 999 999 999 999

Bloc 10

411 11 149 65 218 63 156 718 0 0 202 139 201 728 129 0 707 710 416 999 0 41 140 741 0 999 33 0 0 999 48 0 0 0 0 999 101 999 712 411 0 0 0 999 999 0 0 0 0 0 999 0 0 0 999 42 0 0 0 0 999 0 999 0 999 0 0 0 999 999 0 0 0 0 0 999 0 0 0 999 0 0 0 0 0 999 0 999 0 999 0 0 0 999 999 0 0 0 0 0 999 0 0 0 999 0 0 0 0 0 999 0 999 0 999 0 0 0 999 999 0 0 0 0 0

<u>Recherche harmonique</u>

Bloc 1

204 215 219 202 209 210 201 206 218 205 217 214 207 211 212 203 213 216 220 208 415 13 16 170 159 69 169 70 151 68 168 166 67 167 165 64 35 65 63 42 164 162 158 61 66 999 163 161 650 156 157 160 51 57 62 149 420 999 999 999 154 59 60 58 150 53 419 999 999 999 999 55 220 46 34 56 45 418 999 999 153 999 999 999 999 52 54 50 147 49 25 999 999 999 999 999 148 999 649 38 48 647 999 999 999 204 999 999 999 999 40 33 137 126 43 648 999 999 999 999 999 29 999 645 37

Bloc 2

1411 410 414 19 1 152 750 749 709 747 646 745 748 746 644 155 146 44 143 144 643 743 145 744 642 1401 999 999 641 113 742 741 41 740 638 632 739 127 138 142 141 738 47 140 135 36 139 737 736 640 1415 999 999 39 639 735 635 32 734 636 31 731 136 733 30 732 730 134 702 129 133 132 729 728 218 1414 999 999 26 630 727 28 634 725 726 27 723 722 724 633 721 716 130 720 128 417 999 999 999 999 1410 999 999 416 999 719 624 24 715 718 620 717 714 706 219 713 712 606 711 217 999 999 999 999 999

Bloc 3

1402 8 637 631 626 131 627 629 710 625 628 623 123 622 107 708 125 111 124 619 216 618 621 121 118 1403 21 122 415 999 22 23 117 707 208 116 214 119 112 120 705 617 215 614 115 999 615 613 611 616 1407 114 414 999 999 413 999 999 999 999 411 999 999 999 999 704 604 999 206 412 999 110 212 407 999 1405 408 999 999 999 999 999 999 999 999 999 999 999 999 703 210 999 999 999 999 109 999 999 999 1404 999 999 999 999 999 999 999 999 999 999 999 999 999 999 701 999 999 999 999 999 207 999 999 999

Bloc 4

402 408 4 20 7 612 610 209 607 601 609 106 205 605 404 213 211 105 410 999 603 405 999 999 999 999 999 104 409 999 406 999 999 999 999 403 999 999 999 999 999 999 203 999 999 108 999 999 999 999 999 999 102 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 103 999 999 999 999 999 999 402 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 201 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999

Bloc 5

1215 1206 1204 1203 1212 1202 1216 1211 1209 1201 1205 1220 1217 1213 1214 1219 1218 1208 1207 1210 404 5 18 10 6 101 608 401 999 999 202 602 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0

Bloc 6

1408 405 403 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1406 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1413 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1412 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1409 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Bloc 7

406 15 17 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Bloc 8

401 14 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 9
409 412 407 413 9 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 10
411 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Instance 4

Algorithme génétique

Bloc 1
204 215 219 202 209 210 201 206 218 205 217 214 207 211 212 203 213 216 220 208 415 13 16 116
658 669 63 650 657 652 605 31 629 673 698 601 602 636 610 405 106 668 681 679 682 999 27 190
87 655 0 62 0 642 625 0 0 0 209 186 70 52 611 102 999 0 210 90 660 666 999 0 147 57 623 0 48 0
606 163 0 0 0 999 30 0 0 0 0 999 0 999 83 417 999 999 0 117 0 0 0 0 0 418 999 0 0 0 999 21 0 0 0 0
999 0 999 0 999 999 999 0 0 0 0
Bloc 2
1411 410 414 19 1 667 663 677 637 741 634 685 640 620 635 607 675 695 136 670 91 0 178 614 613
1401 999 999 604 631 612 0 116 630 0 214 101 84 192 56 69 155 146 105 0 78 0 125 187 191 1415
999 999 123 107 608 0 97 626 0 999 0 0 22 55 0 108 135 0 0 0 0 0 414 999 1414 999 999 220 51 205
0 94 109 0 999 0 0 0 0 0 404 999 999 999 0 0 0 999 999 1410 999 999 999 0 999 0 0 0 0 999 0 0 0 0
0 999 999 999 999 0 0 0 999 999
Bloc3
1402 8 0 643 753 770 653 632 198 694 646 654 79 662 177 692 723 157 665 680 80 702 742 622 700
1403 53 0 641 0 0 174 82 119 686 639 140 0 137 100 58 0 92 120 638 40 0 0 188 693 1407 0 0 0 0 0
118 36 86 649 184 0 0 107 67 47 0 0 98 124 0 0 0 0 690 1405 0 0 0 0 0 111 0 77 148 110 0 0 216 0 0
0 0 0 0 0 0 0 0 628 1404 0 0 0 0 0 0 0 45 122 0 0 0 999 0 0 0 0 0 0 0 0 0 0 173
Bloc 4
402 408 4 20 7 674 688 739 624 701 687 627 699 149 33 659 154 706 200 697 199 676 691 678 118
999 999 39 633 664 0 207 0 617 0 648 171 696 0 0 202 0 0 111 0 81 131 0 402 999 999 999 0 609
115 0 999 0 213 0 618 141 645 0 0 999 0 0 217 0 34 24 0 999 999 999 999 0 185 0 0 999 0 999 0 145
105 117 0 0 999 0 0 999 0 0 0 0 999 999 999 999 0 182 0 0 999 0 999 0 120 64 109 0 0 999 0 0 999 0
0 0 0 999 999
Bloc 5
1215 1206 1204 1203 1212 1202 1216 1211 1209 1201 1205 1220 1217 1213 1214 1219 1218 1208
1207 1210 404 5 18 10 6 0 0 108 0 651 0 647 72 46 106 0 656 615 683 684 93 644 616 689 672 999
68 603 114 671 0 0 0 0 211 0 0 43 0 409 0 129 128 0 621 0 0 0 144 661 999 0 112 0 619 0 0 0 0 999
0 0 0 0 999 0 0 114 0 0 0 0 0 23 170 999 0 0 0 420 0 0 0 0 999 0 0 0 0 999 0 0 0 0 0 0 0 0 0 153 999 0
0 0 999
Bloc6
1408 405 403 183 195 28 737 769 0 761 0 172 764 766 204 113 738 76 755 37 722 728 705 181 757
1406 999 999 408 999 0 0 0 0 703 0 169 758 0 999 0 0 44 744 0 0 0 0 88 716 1413 999 999 999 999
0 0 0 0 0 0 119 0 0 999 0 0 42 740 0 0 0 0 0 0 1412 999 999 999 999 0 0 0 0 0 0 50 0 0 999 0 0 0 0 0
0 0 0 0 0 1409 999 999 999 999 0 0 0 0 0 0 41 0 0 999 0 0 0 0 0 0 0 0 0 0
Bloc 7
406 15 17 2 0 99 747 724 704 736 762 765 132 101 760 0 733 752 71 175 0 218 152 167 89 999 35
179 176 0 0 0 720 0 712 0 0 102 0 0 0 0 0 0 406 0 999 215 0 60 999 0 219 403 999 0 0 717 0 0 0 0
412 999 999 0 0 0 0 999 0 999 999 0 0 999 0 999 999 999 0 0 0 0 0 0 0 999 999 999 0 0 0 0 999 0
999 999 0 0 999 0 999 999 999 0 0 0 0 0 0 0 999 999 999 0 0 0 0 999 0 999 999 0 0
Bloc 8

401 14 12 115 735 727 104 767 85 156 0 208 134 32 759 730 709 0 168 768 103 203 165 708 710
999 180 0 411 999 0 401 999 999 999 0 999 126 0 0 0 0 0 73 0 0 999 121 0 0 999 415 999 999 999 0
999 999 999 999 0 999 407 999 999 0 0 0 65 0 0 999 201 0 0 999 999 999 999 999 0 999 999 999
999 0 999 999 999 999 0 0 0 0 0 0 999 999 0 0 999 999 999 999 999 0 999 999 999 999 0 999 999
999 999 0 0 0 0 0 0 999 999 0 0

Bloc 9

409 412 407 413 9 3 160 750 61 166 725 130 103 734 756 718 719 96 715 26 749 721 162 110 743
999 999 999 999 133 194 0 707 59 143 0 0 413 999 999 0 0 54 713 0 0 0 151 75 731 999 999 999
999 104 0 0 0 0 416 0 0 999 999 999 0 0 38 0 0 0 0 138 66 714 999 999 999 999 0 0 0 0 0 999 0 0
999 999 999 0 0 0 0 0 0 0 419 999 999 999 999 999 999 0 0 0 0 0 999 0 0 999 999 999 0 0 0 0 0 0 0
999 999 999

Bloc 10

411 11 0 189 95 748 164 139 726 763 751 196 206 711 754 150 112 142 746 0 729 158 197 0 0
999 29 0 127 0 0 159 0 0 0 732 193 999 0 0 0 0 74 745 0 0 0 0 0 0 999 0 0 113 0 0 212 0 0 0 0 161
999 0 0 0 0 49 0 0 0 0 0 0 0 999 0 0 0 0 0 999 0 0 0 0 410 999 999 999 0 0 25 0 0 0 0 0 0 0 999 0 0 0
0 0 999 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0

Recherche harmonique

Bloc 1

204 215 219 202 209 210 201 206 218 205 217 214 207 211 212 203 213 216 220 208 415 13 16 120
119 117 111 118 115 69 116 96 114 113 199 102 109 73 112 700 107 110 698 108 106 999 105 104
179 103 200 101 198 99 420 699 196 98 697 197 195 97 683 100 95 696 695 194 93 66 999 694 193
92 192 191 190 692 94 999 88 108 693 220 91 86 87 189 691 90 89 690 183 689 182 999 85 134 188
688 187 186 83 684 999 219 185 687 999 184 81 79 686 77 80 84 181 61 178 162 999 78 685 82 750

Bloc 2

1411 410 414 19 1 679 682 681 678 674 180 770 680 676 677 76 672 769 177 673 675 170 75 671
175 1401 999 999 670 119 173 174 172 171 176 70 74 72 169 668 669 71 768 143 667 68 65 660
148 167 1415 999 999 168 636 166 55 67 665 762 419 999 999 999 999 663 64 767 63 666 165 664
164 62 161 1414 999 999 661 54 158 163 120 60 766 999 999 999 999 999 662 57 765 764 156 56
51 53 759 160 1410 999 999 159 58 151 154 153 59 752 999 999 999 999 999 654 49 763 761 659
644 137 24 760 50

Bloc3

1402 8 155 152 658 657 755 651 758 756 656 48 655 218 757 649 648 705 754 652 753 653 751 135
647 1403 157 52 646 47 150 735 747 639 643 650 149 746 999 748 145 418 999 999 999 749 146 45
43 46 1407 645 44 745 744 743 138 742 144 642 42 641 741 999 147 106 999 999 999 999 41 141
417 999 999 1405 640 740 142 40 739 738 140 139 128 637 136 734 999 737 39 999 999 999 999
638 217 999 999 999 1404 635 736 214 35 731 38 36 733 710 732 37 634 999 730 630 999 999 999
999 131 999 999 999 999

Bloc 4

402 408 4 20 7 132 632 633 133 27 34 30 32 33 729 623 29 629 628 129 631 31 626 627 728 999
999 102 126 130 127 28 725 25 26 624 625 727 724 721 125 726 123 122 124 22 621 723 23 622
999 999 114 121 109 21 722 116 620 720 618 118 715 719 209 718 717 416 999 999 117 617 619
716 216 999 999 104 415 999 215 115 112 714 615 213 709 414 999 999 713 613 999 999 999 113
614 207 712 999 999 999 212 999 999 999 711 412 999 999 999 410 999 999 999 608 413 999 999
999 411 999 999 999 999

Bloc 5

1215 1206 1204 1203 1212 1202 1216 1211 1209 1201 1205 1220 1217 1213 1214 1219 1218 1208
1207 1210 404 5 18 10 6 606 616 603 111 211 601 210 611 612 610 110 208 408 999 999 609 409
999 999 999 999 407 999 999 999 206 107 607 406 999 405 999 999 999 999 404 999 999 999 999
205 999 999 999 999 999 999 999 999 999 999 999 105 204 999 999 999 999 999 999 999 999 999 999
999 999 999 999 999 999 999 999 999 999 999 999 999 202 999 999 999 999 999 999 999 999 999 999
999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999

Bloc 6

1408 405 403 707 706 708 704 103 703 403 203 702 402 999 999 701 101 401 999 999 201 0 0 0 0

1406 999 999 0 0 0 0 0 0 999 999 0 999 999 999 0 0 999 999 999 999 0 0 0 0 1413 999 999 0 0 0 0 0 0 999 999 0 999 999 999 0 0 999 999 999 999 0 0 0 0 1412 999 999 0 0 0 0 0 0 999 999 0 999 999 999 0 0 999 999 999 999 0 0 0 0 1409 999 999 0 0 0 0 0 0 999 999 0 999 999 999 0 0 999 999 999 999 0 0 0 0

Bloc 7

406 15 17 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Bloc 8

401 14 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Bloc 9

409 412 407 413 9 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Bloc 10

411 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Instance 5

Algorithme génétique

Bloc 1

204 215 219 202 209 210 201 206 218 205 217 214 207 211 212 203 213 216 220 208 415 13 16 763 774 660 678 680 154 682 101 35 605 173 679 189 80 107 120 683 604 662 661 670 663 999 676 233 0 0 66 667 183 0 618 68 0 0 169 37 206 32 44 120 622 207 650 145 200 124 999 672 88 0 0 59 82 223 0 164 0 0 0 159 0 999 24 0 67 413 999 33 0 403 999 999 657 0 0 0 0 0 999 0 407 0 0 0 153 0 999 0 0 0 999 999 0 0 999 999 999 0 0 0 0

Bloc 2

1411 410 414 19 1 626 198 652 717 684 99 196 690 702 168 697 642 636 647 669 615 637 696 745 691 1401 999 999 0 694 608 62 147 0 412 90 180 617 0 54 614 231 0 149 651 176 119 695 736 0
1415 999 999 0 221 165 30 132 0 999 56 141 106 0 39 195 156 0 112 632 128 63 220 734 0
1414 999 999 0 0 222 0 409 999 999 0 405 999 999 999 75 0 0 0 627 103 0 999 0 0
1410 999 999 0 0 999 0 999 999 999 0 999 999 999 999 70 0 0 0 215 411 999 999 999 999

Bloc 3

1402 8 47 751 613 641 664 685 725 204 133 639 688 677 700 655 665 686 692 61 731 689 719 612 610 1403 0 0 735 234 0 640 654 0 999 0 216 687 102 72 0 143 136 0 0 712 648 0 238 419
1407 0 0 0 135 0 631 425 999 999 0 999 645 23 0 0 0 121 0 0 711 0 0 74 999
1405 0 0 0 408 0 0 999 999 999 0 999 418 999 999 0 0 203 0 0 0 0 0 38 999 1404 0 0 0 999 0 0 999 999 999 0 999 999 999 999 0 0 999 0 0 0 0 0 0 999

Bloc 4

402 408 4 20 7 620 634 759 701 713 699 740 213 0 628 668 698 625 770 671 607 619 709 633 616 999 999 624 0 653 224 151 0 0 0 674 726 999 0 603 0 643 146 0 629 240 51 0 105 223 999 999 611 0 0 57 404 999 999 999 162 0 999 0 421 0 623 415 999 999 194 0 0 0 152 999 999 0 0 0 0 999 999 999 999 225 0 999 0 999 0 601 999 999 999 0 0 0 0 0 999 999 0 0 0 0 999 999 999 999 999 0 999 0 999 0 87 999 999 999 0 0 0 0 0

Bloc 5

1215 1206 1204 1203 1212 1202 1216 1211 1209 1201 1205 1220 1217 1213 1214 1219 1218 1208 1207 1210 404 5 18 10 6 181 117 693 644 111 0 606 675 656 236 646 621 681 114 0 0 0 0 638 414 999 46 160 658 673 114 0 659 102 0 0 602 115 0 161 635 182 666 0 0 0 0 0 0 999 999 0 98 630 422 48 0 649 210 0 0 41 60 0 420 609 71 201 0 0 0 0 0 0 999 999 0 89 0 999 0 0 202 999 0 0 0 0 0 999 140 0 999 0 0 0 0 0 0 999 999 0 79 0 999

Bloc 6

1408 405 403 214 229 746 762 193 126 228 739 239 227 730 119 109 710 130 758 773 55 109 0 92 163 1406 999 999 999 29 0 727 0 116 197 733 94 190 716 402 93 0 69 0 707 0 73 0 0 106 1413 999 999 999 0 0 0 0 49 127 0 0 115 0 999 0 0 43 0 706 0 0 0 0 0 1412 999 999 999 0 0 0 0 0 76 0 0 101 0 999 0 0 0 0 0 0 0 0 0 0 1409 999 999 999 0 0 0 0 0 31 0 0 219 0 999 0 0 0 0 0 0 0 0 0 0

Bloc 7

406 15 17 2 77 756 25 724 772 0 0 184 178 764 138 187 81 729 237 744 714 166 155 750 424 999 226 171 423 999 755 0 0 771 0 0 0 110 0 95 108 0 0 129 0 0 110 131 737 999 999 218 96 999 999 732 0 0 0 0 0 0 84 0 40 36 0 0 0 0 0 27 417 999 999 999 999 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 999 0 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999

Bloc 8

401 14 12 174 42 108 199 52 142 728 107 113 718 150 748 224 222 172 747 123 766 211 749 753 406 999 118 179 410 999 0 167 0 104 0 0 112 0 113 0 999 103 148 0 0 0 999 705 0 999 999 0 139 999 999 0 416 999 999 999 0 21 0 0 0 999 53 221 0 0 0 999 704 0 999 999 0 83 999 999 0 999 999 999 999 0 0 0 0 0 999 0 999 0 0 0 999 0 0 999 999 0 78 999 999 0 999 999 999 999 0 0 0 0 0 999 0 999 0 0 0 999 0 0 999

Bloc 9

409 412 407 413 9 3 157 185 0 721 235 175 64 760 192 117 775 208 743 741 225 761 232 111 708 999 999 999 999 86 144 0 0 0 0 212 0 22 0 118 34 0 999 0 0 186 723 177 97 0 999 999 999 999 0 58 0 0 0 0 999 0 0 0 0 0 0 999 0 0 0 0 209 0 0 999 999 999 999 0 0 0 0 0 0 999 0 0 0 0 0 0 999 0 0 0 0 999 0 0 999 999 999 999 0 0 0 0 0 0 999 0 0 0 0 0 0 999 0 0 0 0 999 0 0

Bloc 10

411 11 765 715 769 768 738 722 28 754 100 122 230 0 105 116 91 742 0 191 170 158 767 45 205 999 134 0 703 0 757 0 720 0 0 0 217 137 0 0 104 50 0 0 188 26 85 752 0 999 999 125 0 0 0 0 0 0 0 0 0 999 0 0 0 65 0 0 0 401 0 0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 999 0 0 0 0 999 999 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 999 0 0 0 0 999

La recherhce harmonique

Bloc1

204 215 219 202 209 210 201 206 218 205 217 214 207 211 212 203 213 216 220 208 415 13 16 238 240 239 237 234 229 180 236 232 113 235 230 233 231 226 120 225 118 119 223 700 227 999 228 425 999 999 87 221 222 224 113 65 117 200 114 112 100 106 116 199 115 104 107 110 109 105 999 111 999 999 999 103 102 101 698 98 699 99 108 198 697 97 673 96 195 197 95 695 89 196 696 999 194 999 999 999 94 93 92 694 193 69 190 175 693 692 91 64 74 88 192 90 189 191 690 188 999 73 999 999 999

Bloc 2

1411 410 414 19 1 689 685 691 688 687 686 186 187 184 684 86 84 681 83 683 185 682 183 182 679 1401 999 999 67 85 82 80 680 79 178 181 179 81 77 665 78 169 678 176 674 675 76 677 75 177 1415 999 999 71 671 773 676 424 999 999 660 72 167 672 70 173 63 423 999 999 174 172 171 170 422 1414 999 999 66 225 772 168 999 999 999 765 669 165 421 999 770 668 999 999 999 68 670 774 775 999 1410 999 999 61 999 771 666 999 999 999 769 761 52 999 999 723 667 999 999 999 60 55 121 163 999

Bloc3

1402 8 664 701 767 762 166 766 62 768 662 764 759 162 657 663 108 739 602 164 731 763 661 656 654 1403 35 760 160 639 161 159 658 158 758 57 659 757 59 156 157 155 756 56 743 152 753 655 755 58 1407 154 54 652 153 224 651 751 754 51 636 653 750 49 650 53 752 45 151 50 749 748 146 149 649 1405 127 747 150 745 999 148 648 48 746 143 646 744 32 147 46 742 43 38 42 44 47 40 647 41 1404 37 139 137 735 999 645 39 741 740 31 142 140 145 144 738 737 733 36 28 644 736 734 134 712

Bloc 4

402 408 4 20 7 643 223 641 642 623 141 638 637 640 622 138 135 633 631 629 420 999 999 999 999 999 999 34 136 132 635 999 131 634 632 133 33 30 419 999 125 732 630 729 130 999 999 999 999 999 999 29 128 129 730 999 27 123 728 25 626 126 999 999 24 727 726 628 26 999 999 999

999 999 999 999 219 627 22 725 999 724 611 216 621 625 222 999 999 124 722 720 21 721 999 999 999 999 999 999 999 999 122 405 221 999 719 119 999 624 718 999 999 999 620 417 999 999 999 999 999 999 999 999

Bloc 5

1215 1206 1204 1203 1212 1202 1216 1211 1209 1201 1205 1220 1217 1213 1214 1219 1218 1208 1207 1210 404 5 18 10 6 23 120 118 117 616 619 618 418 999 999 109 218 116 217 615 115 220 617 215 415 999 114 603 214 614 414 999 999 999 999 413 999 999 999 999 213 999 613 999 112 412 999 999 999 999 999 111 416 999 999 999 999 999 999 999 999 999 999 999 999 999 206 999 608 999 999 999 999 999 999 211 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 203 999 999 999 999 999 999 999 999 999

Bloc 6

1408 405 403 717 716 714 715 713 212 711 409 999 999 999 999 210 411 999 999 999 709 705 110 209 708 1406 999 999 410 999 207 408 999 999 999 999 999 999 999 999 999 999 999 999 999 710 404 999 999 999 1413 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 707 999 999 999 999 1412 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 703 999 999 999 999

Bloc 7

1409 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 706 999 999 999 999 406 15 17 2 208 205 107 406 999 999 201 704 407 999 999 105 104 102 106 103 401 999 999 999 999 999 402 999 999 999 999 204 999 999 999 999 702 999 999 999 202 101 403 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 0 999 999 999 999 0 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 0 999 999 999 999 0 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 0 999 999 999 999 0 999 999 999 999 999 999 999 999

Bloc 8

401 14 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Bloc 9

409 412 407 413 9 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 999 999 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Bloc 10

411 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Annexe 2

Etat initial

Bloc 1

670 170 70 669 668 69 667 102 68 169 666 167 750 168 166 67 664 165 749 65 663 665 64 230 661 66 164 62 157 163 60 430 999 999 999 61 59 63 160 660 158 161 659 748 159 662 129 658 999 652 135 657 58 57 229 162 999 999 999 999 56 55 54 656 50 150 429 999 999 999 51 155 655 999 648 151 653 47 52 999 43 999 999 999 999 48 53 654 49 139 146 999 999 999 999 154 42 212 999 623 602 651 228 156 999 153 999 999 999 999 148 29 18 152 126 225 999 999 999 999 24 2 999 999 415

Bloc 2

649 650 746 428 999 644 147 646 743 647 149 46 745 645 747 144 143 642 141 45 744 44 643 145 741 41 142 641 999 999 39 742 640 740 639 739 638 40 633 635 137 140 101 738 737 35 736 634 637 138 38 628 734 999 999 227 735 136 733 37 636 30 632 134 133 3 729 36 730 731 732 728 34

626 33 226 123 427 999 999 999 32 201 727 631 131 130 404 999 999 25 426 999 999 999 27 725 423 999 999 999 132 999 999 999 999 422 999 999 999 223 629 999 999 999 424 999 999 999 999 421 999 999 999 999

Bloc 3

125 21 627 630 9 723 127 128 26 28 624 625 23 31 124 726 621 122 425 999 121 615 222 120 622 620 220 119 724 221 20 619 419 999 999 420 999 999 999 999 722 19 22 999 999 721 720 999 718 714 219 999 4 719 999 416 999 999 999 999 999 999 999 999 999 217 218 418 999 999 717 117 999 614 716 999 999 118 715 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 413 999 999 999 999 999 999 617 710 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999

Bloc 4

17 618 616 116 414 15 417 999 999 999 216 214 613 713 709 16 14 13 115 114 215 213 211 111 412 12 112 113 411 999 712 999 999 999 999 999 999 11 109 708 8 410 999 999 999 999 999 999 110 999 210 611 409 999 999 610 999 999 999 999 999 999 207 407 999 711 999 999 999 999 999 999 999 209 999 999 408 999 999 999 707 999 999 999 999 999 999 999 999 705 999 999 999 999 999 999 999 999 999 999 999 999 999 999 108 999 999 999 999 999 999 999 999 999 7 999 999 999 999 999 999 999 999 999

Bloc 5

612 609 608 208 607 107 6 606 105 104 106 704 605 604 406 5 706 702 103 10 204 405 999 999 999 205 203 206 999 403 603 1 703 202 224 402 999 999 999 999 601 701 0 401 999 999 999 999 999 999 999 999 999 999 999 0 0 0 999 999 999 999 999 999 0 0 0 999 999 999 999 999 999 999 999 999 999 999 999 0 0 0 999 999 999 999 999 999 0 0 0 999 999 999 999 999 999 999 999 999 999 999 0 0 0 999 999 999 999 999 999 0 0 0 999 999 999 999 999 999

Bloc 6

0 0

Bloc 7

0 0

Bloc 8

0 0

Bloc 9

0 0

Bloc 10

0 0

Instance 1

1ère approche

Bloc 1

670 170 70 669 668 69 667 102 68 169 666 167 750 168 166 67 664 165 749 65 663 665 64 230 661 66 164 62 157 163 60 430 999 999 999 61 59 63 160 660 158 161 659 748 159 662 129 658 999 652 135 657 58 57 229 162 999 999 999 999 56 55 54 656 50 150 429 999 999 999 51 155 655 999 623 151 653 47 52 999 43 999 999 999 999 48 53 654 49 139 146 999 999 999 999 154 42 219 999 648 602 651 228 156 999 153 999 999 999 999 148 29 18 152 126 225 999 999 999 999 24 2 999 999 415

Bloc 2
649 650 746 428 999 644 147 646 743 647 149 46 745 645 747 144 143 642 141 45 744 44 643 145 741 41 142 641 999 999 39 742 640 740 639 739 638 40 633 635 137 140 101 738 737 35 736 634 637 138 38 628 734 999 999 227 735 136 733 37 636 30 632 134 133 3 729 36 730 731 732 728 34 626 33 226 123 427 999 999 999 32 201 727 631 131 130 404 999 999 25 426 999 999 999 27 725 423 999 999 999 132 999 999 999 999 422 999 999 999 223 629 999 999 999 424 999 999 999 999 421 999 999 999 999
Bloc 3
125 21 627 630 9 723 127 128 26 28 624 625 23 31 124 726 621 122 425 999 121 615 222 120 622 620 220 119 724 221 20 619 419 999 999 420 999 999 999 999 722 19 22 999 999 721 720 999 718 714 212 999 4 719 999 416 999 999 999 999 999 999 999 999 999 217 218 418 999 999 717 117 999 614 716 999 999 118 715 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 413 999 999 999 999 999 999 617 710 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999
Bloc 4
17 618 616 116 414 15 417 999 999 999 216 214 613 713 709 16 14 13 115 114 215 213 211 111 412 12 112 113 411 999 712 999 999 999 999 999 999 11 109 708 8 410 999 999 999 999 999 999 110 999 210 611 409 999 999 610 999 999 999 999 999 999 207 407 999 711 999 999 999 999 999 999 999 209 999 999 408 999 999 999 707 999 999 999 999 999 999 999 999 705 999 999 999 999 999 999 999 999 999 999 999 999 999 999 108 999 999 999 999 999 999 999 999 7 999 999 999 999 999 999 999 999 999
Bloc 5
612 609 608 208 607 107 6 606 105 104 106 704 605 604 406 5 706 702 103 10 204 405 999 999 999 205 203 206 999 403 603 1 703 202 224 402 999 999 999 999 601 701 720 401 999 999 999 999 999 999 999 999 999 999 999 97 100 717 999 999 999 999 999 999 999 99 718 703 999 999 999 999 999 999 999 999 999 999 999 999 94 90 719 999 999 999 999 999 999 999 87 707 716 999 999 999 999 999 999 999 999 999 999 999 999 91 73 715 999 999 999 999 999 999 999 215 714 713 999 999 999 999 999 999 999
Bloc 6
96 220 95 82 92 219 93 89 88 86 81 85 98 83 71 80 84 79 78 75 76 130 218 74 72 121 999 120 124 214 999 77 217 216 123 117 106 110 115 213 210 111 129 702 116 208 209 999 212 211 128 999 207 119 999 999 206 999 999 205 103 112 127 204 999 999 203 202 712 126 999 999 999 999 999 125 999 999 201 999 999 999 999 999 999 122 118 114 999 999 999 999 999 711 113 999 999 999 999 999 109 999 999 999 999 999 999 999 999 999 108 107 105 999 999 999 999 999 710 104 999 999 999 999 999
Bloc 7
708 709 705 706 704 701 102 101 0
Bloc 8
0 0
Bloc 9
0 0
Bloc 10
0 0

2ème approche

Bloc 1
670 170 70 669 668 69 667 102 68 169 666 167 750 168 166 67 664 165 749 65 663 665 64 230 661

66 164 62 157 163 60 430 999 999 999 61 59 63 160 660 158 161 659 748 159 662 129 658 999 652
135 657 58 57 229 162 999 999 999 999 56 55 54 656 50 150 429 999 999 999 51 155 655 999 100
151 653 47 52 999 43 999 999 999 999 48 53 654 49 139 146 999 999 999 999 154 42 212 999 95
602 651 228 156 999 153 999 999 999 999 148 29 18 152 126 225 999 999 999 999 24 2 999 999 415
Bloc 2
649 650 746 428 999 644 147 646 743 647 149 46 745 645 747 144 143 642 141 45 744 44 643 145
741 41 142 718 999 999 39 742 640 740 639 739 638 40 633 635 137 140 101 738 737 35 736 634
637 138 38 628 1434 999 999 227 735 136 733 37 636 30 632 134 133 3 729 36 730 731 732 728 34
626 33 91 123 720 1248 99 999 32 201 727 631 131 130 404 999 999 25 426 999 999 999 27 725 423
999 999 426 132 719 130 419 999 422 999 999 999 223 629 999 999 999 424 999 999 999 999 421
999 999 999 999
Bloc 3
125 21 627 630 9 723 127 128 26 28 624 625 23 31 124 726 621 122 425 999 121 615 222 120 622
620 220 119 724 221 20 619 419 999 999 420 999 999 999 999 722 19 22 999 999 721 720 999
718 714 78 999 4 719 999 416 999 999 999 999 999 999 999 999 217 218 418 999 999 717 117
999 614 716 98 999 118 715 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999
413 999 999 999 999 83 999 617 710 999 999 999 999 999 999 999 999 999 999 999 999 999 999
999 999 999 999 999 999 999
Bloc 4
17 618 616 116 414 15 417 999 999 999 216 214 613 713 709 16 14 13 115 114 215 213 211 111 412
12 112 113 411 999 712 999 999 999 999 999 999 11 109 708 8 410 999 999 999 999 999 999 110
999 210 611 409 999 999 610 999 999 999 999 999 999 207 407 999 711 999 999 999 999 999 999
999 209 999 999 408 999 999 999 707 999 999 999 999 999 999 999 999 705 999 999 999 999
999 999 999 999 999 999 999 999 999 999 108 999 999 999 999 999 999 999 999 7 999 999 999
999 999 999 999 999 999
Bloc 5
612 609 608 208 607 107 6 606 105 104 106 704 605 604 406 5 706 702 103 10 204 405 999 999 999
205 203 206 999 403 603 1 703 202 224 402 999 999 999 999 601 701 715 401 999 999 999 999 999
999 999 999 999 999 999 97 94 717 999 999 999 999 999 999 999 1241 716 708 999 999 999 999
999 999 999 999 999 999 999 999 96 86 714 999 999 999 999 999 999 999 90 713 712 999 999 999
999 999 999 999 999 999 999 999 999 75 118 711 999 999 999 999 999 999 999 93 710 709 999 999
999 999 999 999 999
Bloc 6
87 85 74 82 827 92 127 89 88 81 77 79 71 84 126 72 80 220 109 129 219 119 124 73 76 106 212 218
217 999 213 815 999 999 999 108 216 211 115 128 215 112 999 214 125 999 111 202 114 107 209
999 999 999 999 999 999 999 999 999 210 999 999 123 208 999 122 999 999 121 999 120 999 117
116 999 999 999 999 999 999 999 999 999 999 999 999 113 999 999 207 999 999 110 999 105
999 104 206 999 999 999 999 999 999 999 999 999 999 999 999 103 999 999 999 999 999 205
999 204 999 203 999
Bloc 7
702 703 706 201 102 701 101 707 704 705 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 999 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 8
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 9
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 10

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Instance 2

1ère approche

Bloc 1
670 170 62 669 668 69 667 102 68 169 666 167 750 168 166 67 664 165 749 65 663 665 64 230 661 66 164 70 157 163 60 430 999 999 999 61 59 63 160 660 158 161 659 748 159 662 129 658 999 652 135 657 58 57 229 162 999 999 999 999 56 55 54 656 50 150 429 999 999 999 51 155 655 999 623 151 653 47 52 999 43 999 999 999 999 48 53 654 49 139 146 999 999 999 999 154 42 219 999 620 602 651 228 156 999 153 999 999 999 999 148 29 18 152 126 225 999 999 999 999 24 2 999 999 415
Bloc 2
649 650 746 428 999 644 147 646 743 647 149 46 745 645 747 144 143 642 141 45 744 44 643 145 741 41 142 641 999 999 39 742 640 740 639 739 638 40 633 635 137 140 101 738 737 35 736 634 637 138 38 628 734 999 999 227 735 136 733 37 636 30 632 134 133 3 729 36 730 731 732 728 34 626 33 226 123 427 999 999 999 32 201 727 631 131 130 404 999 999 25 426 999 999 999 27 725 423 999 999 999 132 999 999 999 999 422 999 999 999 223 629 999 999 999 424 999 999 999 999 421 999 999 999 999
Bloc 3
125 21 627 630 9 723 127 128 26 28 624 625 23 31 124 726 621 122 425 999 121 615 222 120 622 648 220 119 724 221 20 619 419 999 999 420 999 999 999 999 722 19 22 999 999 721 720 999 718 714 212 999 4 719 999 416 999 999 999 999 999 999 999 999 217 218 418 999 999 717 117 999 614 716 999 999 118 715 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 413 999 999 999 999 999 999 617 710 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999
Bloc 4
17 618 616 116 414 15 417 999 999 999 216 214 613 713 709 16 14 13 115 114 215 213 211 111 412 12 112 113 411 999 712 999 999 999 999 999 999 11 109 708 8 410 999 999 999 999 999 999 110 999 210 611 409 999 999 610 999 999 999 999 999 999 207 407 999 711 999 999 999 999 999 999 999 209 999 999 408 999 999 999 707 999 999 999 999 999 999 999 999 705 999 999 999 999 999 999 999 999 999 999 999 999 999 999 108 999 999 999 999 999 999 999 999 999 7 999 999 999 999 999 999 999 999 999
Bloc 5
612 609 608 208 607 107 6 606 105 104 106 704 605 604 406 5 706 702 103 10 204 405 999 999 999 205 203 206 999 403 603 1 703 202 224 402 999 999 999 999 601 701 719 401 999 999 999 999 999 999 999 999 999 999 999 96 97 715 999 999 999 999 999 999 999 95 720 718 999 999 999 999 999 999 999 999 999 999 999 999 87 85 709 999 999 999 999 999 999 999 77 705 717 999 999 999 999 999 999 999 999 999 999 999 999 130 94 716 999 999 999 999 999 999 999 100 714 713 999 999 999 999 999 999 999
Bloc 6
93 92 90 79 99 98 89 73 712 88 78 81 74 76 72 220 83 86 75 129 71 128 80 91 127
82 122 115 114 215 123 216 84 704 210 219 707 202 105 218 999 126 217 106 125 208 209 119 124 214 213 121 212 211 999 103 999 120 711 999 999 710 999 118 999 999 207 999 206 117 999 999 116 205 999 999 204 999 999 999 203 999 113 708 999 999 706 999 112 999 999 999 999 999 111 999 999 110 999 999 999 999 999 999 999 999 999 109 703 999 999 702 999 201 999 999 999 999 999 108 999 999 107 999 999
Bloc 7
104 701 102 101 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 8

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 9
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 10
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

<u>2ème approche</u>
Bloc 1
670 170 70 669 668 69 667 102 68 169 666 167 750 168 166 67 664 165 749 65 663 665 64 230 661
66 164 93 157 163 60 430 999 999 999 61 59 63 160 660 158 161 659 748 159 662 129 658 999 652
135 657 96 57 229 162 999 999 999 999 56 55 54 656 50 150 429 999 999 999 51 155 655 999 94
151 653 91 52 999 43 999 999 999 999 48 53 654 49 139 146 999 999 999 999 154 42 212 999 90
Bloc 2
602 651 428 156 999 153 999 999 999 999 148 29 18 152 126 225 999 999 999 999 24 2 999 999 89
649 650 746 428 999 644 147 646 743 647 149 46 745 645 747 144 143 642 141 45 744 44 643 145
741 41 142 1432 999 999 39 742 640 740 639 739 638 40 633 635 137 140 101 738 737 35 736 634
637 138 38 628 1434 999 999 227 735 136 733 37 636 30 632 134 133 3 729 36 730 731 1223 728 34
626 33 87 123 1424 130 95 999 32 201 727 631 131 130 404 999 999 25 426 999 999 999 92 725 423
999 999 80 132 1419 827 999 999 422 999 999 999 223 629 999 999 999 424 999 999 999 999 81
718 129 1248 426
Bloc 3
125 21 627 630 9 723 127 128 26 28 624 625 23 31 124 726 621 122 425 999 121 615 222 120 622
72 220 119 88 221 20 619 419 999 999 420 999 999 999 999 722 19 22 999 999 721 720 999 718 714
75 999 4 86 999 416 999 999 999 999 999 999 999 999 999 217 218 418 999 999 717 117 999 614
716 62 999 118 79 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 413 999 999
999 999 82 999 617 58 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999
999 999 999
Bloc 4
17 618 616 116 414 15 417 999 999 999 216 214 613 713 709 16 14 13 115 114 215 213 211 111 412
12 112 113 411 999 712 999 999 999 999 999 999 11 109 708 8 410 999 999 999 999 999 999 110
999 210 611 409 999 999 610 999 999 999 999 999 999 207 407 999 711 999 999 999 999 999 999
999 209 999 999 408 999 999 999 707 999 999 999 999 999 999 999 999 999 705 999 999 999 999
999 999 999 999 999 999 999 999 999 999 108 999 999 999 999 999 999 999 999 7 999 999 999
999 999 999 999 999 999
Bloc 5
612 609 608 208 607 107 6 606 105 104 106 704 605 604 406 5 706 702 103 10 204 405 999 999 999
205 203 206 999 403 603 1 703 202 224 402 999 999 999 999 601 701 720 401 999 999 999 999 999
999 999 999 999 999 999 74 1241 717 999 999 999 999 999 999 999 1220 716 1415 999 999 999 999
999 999 999 999 999 999 999 999 77 127 711 999 999 999 999 999 999 999 85 1410 719 999 999
999 999 999 999 999 999 999 999 999 999 815 999 999 999 999 999 999 999 999 999 62 715 714
999 999 999 999 999 999 999
Bloc 6
84 76 83 27 47 78 73 128 124 419 123 71 58 126 219 125 122 116 121 119 220 120 216 118 821 117
47 115 113 710 106 201 214 218 999 213 27 713 217 999 210 212 114 215 112 999 111 999 202 999
110 211 109 209 712 208 999 999 999 999 999 207 709 999 999 999 999 108 999 206 999 107 999
999 999 205 999 204 999 708 999 999 999 999 999 999 999 707 999 999 999 999 105 999 999 999
203 999 999 999 999 999 999 999 706 999 999 999 999 999 999 705 999 999 999 999 104 999
999 999 999 999 999 999

Bloc 7
103 704 702 101 703 102 701 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 8
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 9
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 10
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Instance 3

1ère approche

Bloc1
670 170 62 669 668 60 667 102 68 169 666 167 750 168 166 67 664 165 749 65 663 665 64 230 661
66 164 70 157 163 69 430 999 999 999 61 59 63 160 660 158 161 659 748 159 662 148 658 999 652
135 657 58 57 229 162 999 999 999 999 56 55 54 656 50 150 429 999 999 999 51 155 655 999 623
151 653 47 52 999 29 999 999 999 999 48 53 654 49 139 146 999 999 999 999 154 42 219 999 620
602 651 228 156 999 153 999 999 999 999 129 43 3 152 126 225 999 999 999 999 24 2 999 999 415
Bloc 2
649 650 746 428 999 644 147 646 743 647 149 46 745 645 747 144 143 642 141 45 744 44 643 145
741 41 142 641 999 999 39 742 640 740 639 739 638 40 633 635 137 140 101 738 731 35 736 634
637 138 38 628 734 999 999 227 735 136 733 37 636 30 632 134 133 18 729 36 730 737 732 728 34
626 33 226 123 427 999 999 999 32 201 727 631 131 130 404 999 999 25 426 999 999 999 27 725
423 999 999 999 132 999 999 999 999 422 999 999 999 223 629 999 999 999 424 999 999 999 999
421 999 999 999 999
Bloc 3
125 21 627 630 9 723 127 128 26 28 624 625 23 31 124 726 621 122 425 999 121 615 222 120 622
648 220 119 724 221 20 619 419 999 999 420 999 999 999 999 722 19 22 999 999 721 720 999 718
714 212 999 4 719 999 416 999 999 999 999 999 999 999 999 999 217 218 418 999 999 717 117 999
614 716 999 999 118 715 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 413
999 999 999 999 999 999 617 710 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999
999 999 999 999 999 999
Bloc 4
17 618 616 116 414 15 417 999 999 999 216 214 613 713 709 16 14 13 115 114 215 213 211 111 412
12 112 113 411 999 712 999 999 999 999 999 999 11 109 708 8 410 999 999 999 999 999 999 110
999 210 611 409 999 999 610 999 999 999 999 999 999 207 407 999 711 999 999 999 999 999 999
999 209 999 999 408 999 999 999 707 999 999 999 999 999 999 999 999 705 999 999 999 999
999 999 999 999 999 999 999 999 999 999 108 999 999 999 999 999 999 999 999 999 7 999 999 999
999 999 999 999 999 999
Bloc 5
612 609 608 208 607 107 6 606 105 104 106 704 605 604 406 5 706 702 103 10 204 405 999 999 999
205 203 206 999 403 603 1 703 202 224 402 999 999 999 999 601 701 720 401 999 999 999 999 999
999 999 999 999 999 999 92 98 718 999 999 999 999 999 999 999 96 716 719 999 999 999 999 999
999 999 999 999 999 999 999 88 91 717 999 999 999 999 999 999 999 97 715 714 999 999 999 999

999 999 999 999 999 999 999 999 75 220 713 999 999 999 999 999 999 999 208 712 711 999 999 999 999 999 999 999

Bloc 6

95 99 89 90 85 86 94 100 84 87 82 93 81 80 74 710 79 77 83 76 78 73 71 126 128 130 116 211 114 218 105 212 101 210 122 219 129 217 124 216 709 209 115 103 72 215 106 110 127 706 125 123 999 214 999 213 999 121 999 120 999 119 999 207 999 708 999 201 206 118 999 117 113 112 707 111 109 999 999 999 999 999 205 999 204 999 108 999 999 999 705 999 999 999 203 999 202 107 104 704 102 0 999 999 999 999 999 999 999 999 999 0 999 999 999 703 999 999 999 999 999 999 0 0 702 701

Bloc 7

0 0

Bloc 8

0 0

Bloc 9

0 0

Bloc 10

0 0

<u>2^{ème} approche</u>

Bloc 1

670 170 70 669 668 69 667 102 68 169 666 167 750 168 166 67 664 165 749 65 663 665 64 230 661 66 164 1223 257 1248 60 430 999 999 999 61 59 63 160 660 158 161 659 748 159 662 129 658 999 652 135 657 92 84 62 162 999 999 999 999 56 55 54 656 50 150 429 999 999 999 51 155 655 999 90 151 653 81 91 429 43 999 999 999 999 48 53 654 49 139 146 999 999 999 999 154 42 212 999 89 602 651 827 999 999 153 999 999 999 999 148 83 88 152 126 225 999 999 999 999 24 2 999 999 263

Bloc 2

649 650 746 428 999 644 147 646 743 647 149 46 745 645 747 144 143 642 141 45 744 44 643 145 741 41 142 720 999 999 39 742 640 740 639 739 638 40 633 635 137 140 101 738 737 35 736 634 637 138 38 628 1434 999 999 227 735 136 733 37 636 30 632 134 133 3 729 36 730 717 78 728 34 626 33 87 123 1432 428 821 999 32 201 727 631 131 130 404 999 999 25 426 999 999 999 76 725 423 999 999 18 132 1431 999 999 999 422 999 999 999 223 629 999 999 999 424 999 999 999 999 72 1424 1241 1220 426

Bloc 3

125 21 627 630 9 723 127 128 26 28 624 625 23 31 124 726 621 122 425 999 121 615 222 120 622 86 220 119 82 221 20 619 419 999 999 420 999 999 999 999 722 19 22 999 999 721 720 999 718 714 85 999 4 80 999 416 999 999 999 999 999 999 999 999 999 217 218 418 999 999 717 117 999 614 716 256 999 118 73 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 413 999 999 999 999 130 999 617 815 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999

Bloc 4

17 618 616 116 414 15 417 999 999 999 216 214 613 713 709 16 14 13 115 114 215 213 211 111 412 12 112 113 411 999 712 999 999 999 999 999 999 11 109 708 8 410 999 999 999 999 999 999 110 999 210 611 409 999 999 610 999 999 999 999 999 999 207 407 999 711 999 999 999 999 999 999 999 209 999 999 408 999 999 999 707 999 999 999 999 999 999 999 999 705 999 999 999 999

999 999 999 999 999 999 999 999 999 999 108 999 999 999 999 999 999 999 999 999 7 999 999 999
999 999 999 999 999 999
Bloc 5
612 609 608 208 607 107 6 606 105 104 106 704 605 604 406 5 706 702 103 10 204 405 999 999 999
205 203 206 999 403 603 1 703 202 224 402 999 999 999 999 601 701 715 401 999 999 999 999 999
999 999 999 999 999 999 79 77 718 999 999 999 999 999 999 999 29 716 710 999 999 999 999 999
999 999 999 999 999 999 999 52 58 1415 999 999 999 999 999 999 999 122 714 1419 999 999 999
999 999 999 999 999 999 999 999 999 218 213 709 999 999 999 999 999 999 999 52 1410 719 999
999 999 999 999 999 999
Bloc 6
57 47 74 75 127 126 27 121 128 129 119 125 124 419 71 123 120 216 118 116 220 217 214 115 713
117 219 109 706 106 210 108 201 215 212 62 207 112 999 101 58 208 999 57 211 999 999 999 209
712 114 999 113 711 111 999 47 999 999 999 206 999 110 999 107 205 999 999 29 999 999 999 999
999 708 27 999 204 707 203 999 18 999 999 999 999 999 105 999 104 999 999 999 202 999 999 999
999 999 705 103 999 999 704 999 999 102 999 999 999 999 999 0 999 0 999 999 999 999 999 999
999 999 999 703
Bloc 7
702 701 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 8
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 9
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 10
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Instance 4

1ère approche

Bloc1
670 170 62 669 668 60 667 102 68 169 666 167 750 168 166 67 664 165 749 65 663 665 64 230 661
66 164 70 157 163 69 430 999 999 999 61 59 63 160 660 158 161 659 748 159 662 148 658 999 652
152 657 58 57 229 162 999 999 999 999 56 55 54 656 50 150 429 999 999 999 51 155 655 999 623
151 653 47 52 999 33 999 999 999 999 48 53 654 49 139 146 999 999 999 999 154 42 219 999 620
602 651 228 156 999 153 999 999 999 999 129 43 3 135 132 225 999 999 999 999 36 2 999 999 415
Bloc 2
649 650 746 428 999 644 147 646 743 647 149 46 745 645 747 144 143 642 141 45 744 44 643 145
741 41 142 641 999 999 39 742 640 740 639 739 638 40 633 635 137 140 101 738 731 35 736 634
637 138 38 628 734 999 999 227 735 136 733 37 636 30 632 134 133 18 729 24 730 737 732 728 34
626 29 226 123 427 999 999 999 32 201 727 631 131 130 404 999 999 25 426 999 999 999 27 725
423 999 999 999 126 999 999 999 999 422 999 999 999 223 629 999 999 999 424 999 999 999 999
421 999 999 999 999
Bloc 3
125 21 627 630 9 723 127 128 26 28 624 625 23 31 124 726 621 122 425 999 121 615 222 120 622
648 220 119 724 221 20 619 419 999 999 420 999 999 999 999 722 19 22 999 999 721 720 999 718
714 212 999 4 719 999 416 999 999 999 999 999 999 999 999 999 217 218 418 999 999 717 117 999

614 716 999 999 118 715 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 413
999 999 999 999 999 999 617 710 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999
999 999 999 999 999 999
Bloc 4
17 618 616 116 414 15 417 999 999 999 216 214 613 713 709 16 14 13 115 114 215 213 211 111 412
12 112 113 411 999 712 999 999 999 999 999 999 11 109 708 8 410 999 999 999 999 999 999 110
999 210 611 409 999 999 610 999 999 999 999 999 207 407 999 711 999 999 999 999 999 999
999 209 999 999 408 999 999 999 707 999 999 999 999 999 999 999 999 999 705 999 999 999 999
999 999 999 999 999 999 999 999 999 999 108 999 999 999 999 999 999 999 999 7 999 999 999
999 999 999 999 999 999
Bloc 5
612 609 608 208 607 107 6 606 105 104 106 704 605 604 406 5 706 702 103 10 204 405 999 999 999
205 203 206 999 403 603 1 703 202 224 402 999 999 999 999 601 701 720 401 999 999 999 999 999
999 999 999 999 999 999 92 100 717 999 999 999 999 999 999 999 97 719 718 999 999 999 999 999
999 999 999 999 999 999 999 94 96 716 999 999 999 999 999 999 999 91 715 714 999 999 999 999
999 999 999 999 999 999 999 999 214 99 713 999 999 999 999 999 999 999 78 712 711 999 999 999
999 999 999 999
Bloc 6
98 88 95 79 89 90 87 93 84 83 82 86 75 81 74 120 85 72 77 80 123 219 73 76 71 215 130 211 220
117 107 119 218 128 116 129 122 206 212 109 707 216 105 111 204 127 999 203 124 205 999 126
999 999 202 217 125 999 121 118 102 115 999 999 114 705 999 113 112 999 110 999 999 108 999
999 213 999 999 999 999 210 999 106 209 208 104 999 999 207 710 999 103 101 999 201 999 999 0
999 999 999 999 999 999 999 999 999 0 999 999 0 999 999 999 709 999 0 0 999 999 999 999 0 999
Bloc 7
706 704 702 708 703 701 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 8
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 9
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bloc 10
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

2ème approche

Bloc1
670 170 70 669 668 69 667 102 68 169 666 167 750 168 166 67 664 165 749 65 663 665 64 230 661
66 164 1223 157 163 85 430 999 999 999 61 59 63 160 660 158 161 659 748 159 662 129 658 999
652 135 657 81 57 229 86 999 999 999 999 56 55 54 656 50 150 429 999 999 999 51 155 655 999 89
151 653 74 52 999 83 999 999 999 999 48 53 654 49 139 146 999 999 999 999 154 42 212 999 79
602 651 62 156 999 80 999 999 999 999 84 87 1248 72 118 225 999 999 999 999 24 2 999 999 213
Bloc 2
649 650 746 428 999 644 147 646 743 647 149 46 745 645 747 144 143 642 141 45 744 44 643 145
741 41 142 720 999 999 39 742 640 740 639 739 638 40 633 635 137 140 101 738 737 35 736 634
637 138 38 628 1434 999 999 227 735 136 733 37 636 30 632 134 88 76 729 428 730 1432 60 728 34
626 33 18 123 1431 82 78 999 32 201 727 631 131 130 404 999 999 73 426 999 999 999 248 725 423

999 999 77 132 1424 827 999 999 422 999 999 999 223 629 999 999 999 25 824 999 999 999 29 715 252 821 999

Bloc 3

125 21 627 630 9 723 127 128 26 28 624 625 23 31 124 726 621 122 425 999 121 615 222 120 622 75 220 119 71 221 20 619 419 999 999 420 999 999 999 999 722 19 22 999 999 721 720 999 718 714 110 999 4 47 999 416 999 999 999 999 999 999 999 999 999 217 218 418 999 999 717 117 999 614 716 1241 999 118 219 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 413 999 999 999 999 62 999 617 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999

Bloc 4

17 618 616 116 414 15 417 999 999 999 216 214 613 713 709 16 14 13 115 114 215 213 211 111 412 12 112 113 411 999 712 999 999 999 999 999 999 11 109 708 8 410 999 999 999 999 999 999 110 999 210 611 409 999 999 610 999 999 999 999 999 999 207 407 999 711 999 999 999 999 999 999 999 209 999 999 408 999 999 999 707 999 999 999 999 999 999 999 999 705 999 999 999 999 999 999 999 999 999 999 999 999 999 999 108 999 999 999 999 999 999 999 999 7 999 999 999 999 999 999 999 999 999

Bloc 5

612 609 608 208 607 107 6 606 105 104 106 704 605 604 406 5 706 702 103 10 204 405 999 999 999 205 203 206 999 403 603 1 703 202 224 402 999 999 999 999 601 701 1419 401 999 999 999 999 999 999 999 999 999 999 999 426 3 719 999 999 999 999 999 999 999 58 707 1415 999 999 999 999 999 999 999 999 999 999 999 999 999 36 1410 999 999 999 999 999 999 999 27 705 718 999 999 999 999 999 999 999 999 999 999 999 999 999 1220 717 999 999 999 999 999 999 999 43 716 714 999 999 999 999 999 999 999

Bloc 6

60 226 815 999 999 103 262 58 419 233 124 253 128 130 101 126 123 122 129 220 47 218 127 121 105 211 210 999 999 999 119 205 125 999 115 117 217 203 108 43 106 36 114 29 999 212 999 215 204 216 999 999 999 999 999 27 999 25 999 120 18 999 999 116 3 214 201 209 113 999 999 999 999 999 999 999 999 999 999 999 208 999 207 999 112 111 999 999 206 109 999 999 999 202 999 999 999 999 999 999 999 999 999 999 999 999 999 999 999 107 104 999 999 999 102 999 999 999 999 999 999 999 999 999 999

Bloc 7

711 713 710 712 709 708 706 704 701 703 702 0

Bloc 8

0 0

Bloc 9

0 0

Bloc 10

0 0

Bibliographie

Dans l'ensemble, les livres furent son expérience.
Henri Michaux

[Al-Betar et al., 2008] Al-Betar, M., Khader, A., and Gani, T. (2008). A harmony search algorithm for university course timetabling. In 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, Canada.

[Avriel et col., 1993] Avriel, M. et Penn, M., Exact and approximate solutions of the container ship stowage problem, Computers and Industrial Engineering, pp. 271–274, 1993.

[Avriel et col., 1998] Avriel M., Penn M., shiprer N. et Witteboon S., Stowage planning for container ships to reduce the number of shifts, Annals of Operations Research 76, p. 55-71, 1998

[Bazzazi et col., 2009] Bazzazi, M., Safaei, N. and Javadian, N., 2009, A genetic algorithm to solve the storage space allocation problem in a container terminal, Computers & Industrial Engineering 36 (2009), p. 1711–1725.

[Berro, 2001] Alain Berro, Optimisation multiobjectif et stratégies d'évolution en environnement dynamique, thèse en informatique soutenu le 18 décembre 2001, Université des Sciences Sociales Toulouse I

[Blythe, 1999] Blythe Jim, Decision-Theoretic Planning, AI Magazine, Summer 1999

[Borne et col., 1990] P. Borne, G.D. Tanguy, J.P. Richard, F. Rotella, I. Zambettakis, "Commande et optimisation de processus", Edition Technip, France, 1990.

[Banke , 2001] Jürgen Branke, *Evolutionary Approaches to Dynamic Optmization Problems - Updated survey*, GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, p.27-30, 2001.

[Castro, 1999] Castro, C. F. d. (1999). Containerization, Logistic Cost and Facilitation (less documented aspects of an old theme). In *Regional transport and transit facilitation workshop*. Bangkok.

[Chen et col., 2000] Chen, C.Y., Chao, S.L., Hsieh, T.W., 2000. A time-space network model for the space resource allocation problem in container marine transportation, paper presented at the 17th international symposium on mathematical programming 2000, Atlanta, USA.

[Chen et col., 2007] Chen Lu, Nathalie Bostel, Pierre Dejax, Jianguo Cai et Lifeng Xi, A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal, European Journal of Operational Research, Volume 181, Issue 1, 16 August 2007, Pages 40-58

[Chen et col., 2004] Ping Chen, Zhaohui Fu, Lim, A., Rodrigues, B., Port yard storage optimization, Automation Science and Engineering, IEEE Transactions on, 2004, Volume: 1, pp 26 – 37

[Corry et Kozan, 2006] Corry, P. and E. Kozan. (2006). An assignment model for dynamic load planning of intermodal trains. *Computers and Operations Research*, 33: 1-17.

[Davis, 1991] L. Davis , The genetic Algorithm Handbook. Ed. New-York: Van Nostrand Reinhold, ch.17, 1991.

[De Castilho et Daganzo, 1993] De Castilho, B., Daganzo, C.F., 1993. Handling Strategies for import containers at Marine Terminals. Transportation Research B 27 (2), 151–166.

[Dubreuil, 2008] Dubreuil Julien, 2008, La logistique des terminaux portuaires de conteneurs, Centre interuniversitaire de recherche sur les réseaux d'entreprise, la logistique et le transport (CIRRELT) et Département de management et technologie, Université du Québec à Montréal

[Erdal and Saka, 2008] Erdal, F. and Saka, M. (2008). E_ect of beam spacing in the harmony search based optimum design of grillages. Asian Journal of Civil Engineering (Building and Housing), 9(3):215-228.

[Coffman et col.,1983] E. G. Coffman, Jr., M. R. Garey, and D. S. Johnson. Dynamic bin packing. SIAM Journal on Computing, 12(2):227–258, 1983.

[Fesanghary et al., 2008] Fesanghary, M., Mahdavi, M., Minary-Jolandan, M., and Alizadeh, Y. (2008). Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems. Computer Methods in Applied Mechanics and Engineering, 197(33-40):3080-3091.

[Galuszka, 2011] A. Galuszka, Transforming a STRIPS-like Planning to Linear Programming for Re-

handling Operations in Container Terminals, ICGST AIML-11 Conference, Dubai, UAE, 12-14 April 2011

[Geem et al., 2000] Geem, Z., Kim, J., and Yoon, Y. (2000). Optimal layout of pipe networks using harmony search. In Proceedings of 4th International Conference on Hydro-Science and Engineering, Seoul, South Korea.

[Geem et al., 2002] Geem, Z., Kim, J., and Loganathan, G. (2002). Harmony search optimization: Application to pipe network design. International Journal of Modelling & Simulation, 22(2):125-133.

[Geem, 2008] Geem, Z. (2008). Harmony search optimisation to the pump-included water distribution network design. Civil Engineering and Environmental Systems,.

[Geem, 2007] Geem, Z. (2007). Harmony search algorithm for the optimal design of largescale water distribution network. In Proceedings of the 7th International IWA Symposium on Systems Analysis and Integrated Assessment in Water Management (Watermatex 2007), IWA, Washington DC, USA (2007)

[Geem, 2005a] Geem, Z. (2005a). School bus routing using harmony search. In Genetic and Evolutionary Computation Conference (GECCO 2005), Washington DC, USA.

[Geem et al., 2005b] Geem, Z., Tseng, C., and Park, Y. (2005b). Harmony search for generalized orienteering problem: best touring in China. Lecture Notes in Computer Science, pp612:741.

[Geem, 2006] Geem, Z. (2006a). Comparison Harmony Search with Other Meta-Heuristics in Water Distribution Network Design. ASCE.

[Geem, 2000] Geem, Z. (2000). Optimal design of water distribution networks using harmony search. PhD thesis, Korea University.

[Geem and Hwangbo, 2006] Geem, Z. and Hwangbo, H. (2006). Application of harmony search to multi-objective optimization for satellite heat pipe design. In Proceedings of US-Korea Conference on Science, Technology, and Entrepreneurship (UKC 2006), Teaneck, NJ, USA

[Goldberg et Lingle, 1985] Goldberg, D. E. and Lingle, R. (1985), Alleles, loci and the traveling salesman problem, *Proceedings of an International Conference on Genetic Algorithms*, pages 10-19. Morgan Kauffman.

[Gendron et Crainic Teodor , 1995] Gendron Bernard **,** Crainic Teodor Gabriel **,** A branch-and-bound algorithm for depot location and container fleet management, Location Science, Volume 3, Issue 1, May 1995, Pages 39-53

[Henesey, 2004] Henesey, L, A Multi Agent Based Simulator for Managing a Container Terminal, Proceedings of the 2nd European Workshop on Multi-Agent Systems (EUMAS 2004), Barcelona, Spain, pp. 291-302, December 16 &17, 2004.

[Imai et Chen, 2008] Imai, A., Chen H.C., Nishimura, E., Papadimitriu, S. (2008). The simultaneous berth and quay crane allocation problem. *Transportation Research Part E 44,* 900-920

[Imai et col., 1997] Imai, A., Nagaiwa, K., Tat, C.W., 1997, Efficient planning of berth allocation for container terminals in Asia. Journal of Advanced Transportation 31 (1), 75–94.

[Imai et col., 2006] Imai, A., Sasaki, K., Nishimura, E. and Papadimitriou, S., Multi-objective simultaneous stowage and load planning for a container ship with container rehandle in yard stacks, European Journal of Operational Research, 2006

[Holland, 1975] J. H. Holland, *Adaptation In Natural And Artificial Systems*, University of Michigan Press (1975)

[Ioannou et Jula,2008] Ioannou, P. et Jula, H. (2008), Automated Container Terminal Concepts. In P. Ioannou (Ed.), *Intelligent Freight Transportation*. (pp. 7-33). Taylor & Francis Group.

[Jenkins , 1992] Jenkins,W. M. (1992). "Plane Frame Optimum Design Environment Based on Genetic Algorithm." *J. of Structural Engng*, ASCE, 118, 3103.

[Jin, 2010] Jin Li Hong-xia Yu, A Mathematical Model for Vehicle Dispatching Problems at Marine Container Terminals , Intelligent System Design and Engineering Application (ISDEA), 2010, VOL. 1 , PAGE. 63 - 65 , CHANGSHA

[Jiang et col., 2011] Wei Jiang, Yun Dong, Lixin Tang , Logistics Inst., Northeastern Univ., Shenyang, China, Service Operations, Logistics, and Informatics (SOLI), 2011 IEEE International Conference.

[Kammarti et col. 2004] R. Kammarti, S. Hammadi, P. Borne, M. Ksouri, "A new hybrid evolutionary approach for the pickup and delivery problem with time windows". IEEE International Conference on Systems, Man and Cybernetic. 2004. Volume 2, P 1498-1503, Oct 2004

[Kammarti et col. 2005a] R. Kammarti, S. Hammadi, P. Borne, M. Ksouri, "Improved tabu search in an hybrid evolutionary approach for the pickup and delivery problem with time windows", Intelligent Transportation Systems, 2005. Proceeding 2005 IEEE, p 148- 153, 2005.

[Kammarti et col. 2005b] Kammarti R. , S. Hammadi, P. Borne, M. Ksouri, "Lower Bounds In An Hybrid Evolutionary Approach For The Pickup And Delivery Problem With Time Windows". IEEE International Conference on Systems, Man and Cybernetics. 2005. Volume 2, P 1156-1161, Oct 2005.

[Kammarti, 2006] R. Kammarti, Approches évolutionnistes pour la résolution du 1-PDPTW Statique et dynamique, Thèse en automatique et informatique industrielle à l'école centrale de Lille et l'université de Lille, Soutenue le 13 décembre 2006

[Kang et col., 2006][a] Kang, J., Ryu, K.R., Kim, K.H., 2006[a]. Determination of storage locations for incoming containers of uncertain weight. In: Ali, M., Dapoigny, R. (Eds.). Advances in applied artificial intelligence. Proceedings of the 19th Interna- tional Conference on Industrial, Engineering and other Applications of Applied Intelligent Systems, IEA/AIE 2006, vol. 4031. Annecy, France, June 27–30, Springer. pp. 1159–1168.

[Kang et col., 2006][b] Kang, J., Ryu, K.R., Kim, K.H., 2006 [b], Deriving stacking strategies for export containers with uncertain weight information. Journal of Intelligent Manufacturing , volume 17, pp 399–410.

[Kefi, 2008] Kefi Gazdar Meriam, Optimisation Heuristique Distribuée du Problème de Stockage de Conteneurs dans un Port, thèse soutenue le 23 Juin 2008.

[Kefi et col., 2007] M. Kefi, O. Korbaa, K. Ghédira, and P. Yim. *Agent and Multi-Agent Systems : Technologies and Applications*, volume 4496/2007 of *Lecture Notes in Computer Science*, chapter Container Handling using Multi-Agent Architecture, pages 685–693. 2007.

[Kim et Moon, 2003] Kim, K.H., & Moon, K.C. (2003). Berth scheduling by simulated annealing. *Transportation Research Part B 37,* 541–560.

[Kim , 2008] Kim, K.H. (2008). Operational Issues in Modern Container Terminals. In P. Ioannou (Ed). *Intelligent Freight Transportation.* (pp. 51-69). Taylor & Francis Group.

[Kim et Park, 2003] Kim, K-H., Park, K-T., 2003, A note on a dynamic space-allocation method for outbound containers, European Journal of Operational Research, p. 92–101

[Kim, 1997] Kim, K-H., 1997, Evaluation of the number of rehandles in container yards, Computers ind. Engng Vol. 32, No. 4

[Kim et Kim, 1998] Kim, K-H. et Kim, H-B., 1998, The optimal determination of the space requirement and the number of transfer cranes for import containers, Computers ind. Engng Vol. 35, p. 427-430

[Kim et col., 2000] Kim K.H, Park Y.M, Ryu K.R, Deriving decision rules to locate export containers in container yards, European Journal of Operational Research 124 (2000), pp 89-101

[Kim et Bae, 2007] Kim, K.H., Bae, J.W., 1998, Re-marshaling export containers in port container terminals, C&IE, 35, 655- 658

[Kim et col., 2007] Kim, Y.H., Park T., Ryu, K.R., 2007, Dynamic weight adjustment for developing a stacking policy for automated container terminals. Paper presented at The International Conference on Intelligent Manufacturing and Logistics Systems (IML 2007), Kitakyushu, Japan, 26–28 February 2007.

[Korbaa et Yim, 2004] Korbaa, O., Yim, P., 2004, Container Assignment to Stock in a Fluvial Port, S. M. C International Conf., The Netherlands.

[Kozan, 2000] Kozan, E. (2000). Optimising Container Transfers at Multimodal Terminals. *Mathematical and Computer Modelling*, 31: 235-243.

[Kozan et Casey, 2006] Kozan, E. and B. Casey. (2006). Alternative algorithms for the optimization of a simulation model of a multimodal container terminal. *Journal of the Operational Research Society*.

[Liang et Geem, 2005] Lee K.S et Geem Z.W, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, Comput. Methods Appl. Mech. Engrg. 194 (2005) 3902–3933

[Lee et col., 2009] Lee, D.-H., J. X. Cao, Q. Shi and J. H. Chen (2009) A heuristic algorithm for yard truck scheduling and storage allocation problems, *Transportation Research Part E*, 45 (5) 810–820.

[Lee et col., 2005] Lee Yong Hwan , Kang Jaeho , Ryu Kwang Ryel et Kim Kap Hwan, Optimization of Container Load Sequencing by a Hybrid of Ant Colony Optimization and Tabu Search, ADVANCES IN NATURAL COMPUTATION, 2005, Volume 3611/2005.

[Lee et Chao, 2009] Yusin Lee , Shih-Liang Chao, A neighborhood search heuristic for pre-marshalling export containers, Production, Manufacturing and Logistics, European Journal of Operational Research, Volume 196, Issue 2, 16 July 2009, Pages 468-475.

[Liang et col., 2007] Liang Shyi-Ching , Lee Chi-Yu , Huang Shih-Wei , A Hybrid Meta-heuristic for the Container Loading Problem , Communications of the IIMA / Dec, 2007

[Liang et col., 2009] Liang, C., Huang, Y., Yang, Y. (2009). A quay crane dynamic scheduling problem by hybrid evolutionary algorithm for berth allocation planning. *Computers & Industrial Engineering 56*, 1021-1028.

[Macharis et col., 2004] Macharis, C. and Y. M. Bontekoning. (2004). Opportunities for OR in intermodal freight transport research: A review. *European Journal of Operational Research*, 153: 400-416.

[Mahdavi et col., 2007] Mahdavi, M., Fesanghary, M., and Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. Applied Mathematics and Computation, 188(2):1567-1579.

[Meersman, 2002] Meersman P.J.M. (2002). Optimization of Container Handling Systems. *Ph.D. thesis, Erasmus University Rotterdam.*

[Meersmans et Wagelmans, 2000] Patrick J.M. Meersmans et Albert P.M. Wagelmans**,** Dynamic scheduling of handling equipment at automated container terminals, Research Paper from Erasmus Research Institute of Management (ERIM), , Erasmus , 2000, University and the Erasmus School of Economics (ESE) at Erasmus University Rotterdam.

[Mesghouni, 1999] Mesghouni, K., Application des algorithmes évolutionnistes dans les problèmes d'optimisation en ordonnancement de production". Thèse en automatique et informatique industrielle à l'école centrale de Lille et l'université de Lille1. Soutenue le 5 Janvier 1999.

[Mehta et col., 2006] Mehta Ashish, Smith Jay, H. J. Siegel, Anthony A. Maciejewski, Arun Jayaseelan, and Bin Ye Dynamic Resource Management Heuristics for Minimizing Makespan while Maintaining an Acceptable Level of Robustness in an Uncertain Environment, 12th International Conference on Parallel and Distributed Systems (2006), Minneapolis, Minnesota, USA

[Omran and Mahdavi, 2008] Omran, M. and Mahdavi, M. (2008). Global-best harmony search. Applied Mathematics and Computation, 198(2):643-656.

[Park et col., 2003] Park, Y.M., Kim, K.H. (2003). A scheduling method for berth and quay cranes. *OR Spectrum,* 25, 1–23.

[Park et col., 2011] Park T., Choe R., Kim, Y.H., Ryu, K.R., 2011, Dynamic adjustment of container stacking policy in an automated container terminal, Int. J. Production Economics, pp 385–392

[Park et col., 2009] Park, K., Park, T., Ryu, K.R., 2009, Planning for remarshaling in an automated container terminal using cooperative coevolutionary algorithm, Proceeding of the 2009 ACM Symposium on Applied Computing, SAC'09. ACM, New York, NY, pp. 1098–1105.

[Ryu et al., 2007] Ryu, S., Duggal, A., Heyl, C., and Geem, Z. (2007). Mooring cost optimization via

harmony search. In Proceedings of the 26th ASME International Conference on Offshore Mechanics and Arctic Engineering.

[Sciomachen et col., 2007] Sciomachen, A. and Tanfani, E., A 3D-BPP approach for optimising stowage plans and terminal productivity, European Journal of Operational Research, Volume 183, Issue 3, , Pages 1433-1446, 16 December 2007

[Shields, 1984] Shields, J.J., 1984. Container Stowage: A computer aided preplanning system. Marine Technology 21 (4).

[Steenken et col., 2004] Steenken D., Voß S., and Stahlbock R. 2004, Container terminal operation and operations research – a classification and literature review. OR Spectrum (2004) 26: 3–49

[Tfaili, 2007] Walid Tfaili, Conception d'un algorithme de colonie de fourmis pour l'optimisation continue dynamique, thèse en science de l'ingénieur (optimisation), soutenue en décembre 2007 à l'université de Paris 12

[UNCTAD, 2009] UNCTAD (2009). Review of Maritime Transport. United Nations, New York and Geneva.

[Vander, 2000] Vander Meer, R., 2000. Operational control of internal transport, ERIM Ph.D. series Research in Management

[Vis et Koster, 2003] Vis I.F.A. et Koster R. (2003). Transshipment of containers at a container terminal: An overview. *European journal of operational research 147,* 1-16.

[Weiss, 1999] G. Weiss. Multi-agent systems: a modern approach to distributed artificial intelligence. MIT Press, Massachusetts, 1999.

[Wilson et Roach, 2000] Wilson, I.D., Roach, P.A., 2000. Container stowage planning: a methodology for generating computerised solutions. Journal of the Operational Research Society 51, 1248–1255.

[Xiong et Jutan, 2003] Q. Xiong and A. Jutan. Continuous optimization using a dynamic simplex method. Chemical Engineering Science, 58(16) :3817–3828, 2003.

[Zhang et col., 2003] Zhang, C., Liu, J., Wan, Y-W., Murty, K-G. and Linn, R-J., 2003, Storage space allocation in container terminals, Transportation Research Part B, 37, p. 883–903.

[Zhang et col., 2001] Chuqian Zhang, Jiyin Liu, Yat wah Wan, Katta G. Murty, and Richard J. Linn. Storage space allocation in container terminals. Transportation Research (B), 2001.

Techniques avancées d'optimisation pour la résolution du Problème de Stockage de Conteneurs dans un Port

Résumé

Le chargement/déchargement des conteneurs et leurs stockages provisoires dans le port est la plus importante et complexe tâche dans les terminaux portuaires. Elle est fortement liée au routage des grues de quai et son coût augmente considérablement surtout en absence d'une gestion efficace du terminal.
Dans ce travail, nous étudions le problème de stockage des conteneurs (PSC). Il appartient à la catégorie des problèmes NP-difficiles et NP-complets. PSC consiste à déterminer un plan d'arrangement des conteneurs destinés à l'import et à l'export dans le port qui minimise les remaniements ultérieurs lors de leur transfert vers le bateau, camion ou train. En effet, le temps d'attente des camions des clients, le temps de transfert des grues de quai et le temps nécessaire au chargement/déchargement du navire sont avantageusement réduits.
PSC est généralement étudié en considérant un seul type de conteneur. Cependant, plusieurs types de conteneurs sont utilisés dans les ports maritimes (dry, réfrigérés, toit ouvert,...). En outre, le problème de stockage de conteneurs peut être traité de façon statique ou dynamique (date d'arrivée et de départ des conteneurs incertains).
L'objectif de cette thèse est de résoudre le PSC statique et le PSC dynamique pour un seul et plusieurs types de conteneurs en utilisant deux métaheuristiques : l'algorithme génétique, la recherche harmonique
Pour vérifier la performance de chacune des approches proposées, une étude comparative des résultats générés par chaque méthode ainsi que celle de l'algorithme LIFO est établie.

Mots clés

Problème de stockage des conteneurs, statique, dynamique, mouvements de remaniements, métaheuristiques, algorithme génétique, algorithme de la recherché harmonique, optimisation.

Advanced optimization techniques for solving the containers storage problem

Abstract:

The loading and unloading of containers and their temporary storage in the container terminal are the most important and complex operation in seaport terminals. It is highly inter-related with the routing of yard crane and truck and their costs increased significantly especially without an efficient terminal management. To improve this process, an efficiency decision for the container storage space allocation must be taken.

In this thesis, we studied the container storage problem (CSP). It falls into the category of NP hard and NP complete problems. CSP consists on finding the most suitable storage location for incoming containers that minimizes rehandling operations of containers during their transfer to the ship, truck or train. In fact, the wait time of customer trucks, the transfer time of yard crane and the Ship turnaround time are advantageously reduced.

Generally, this problem is studied considering a single container type. However, this does not stand the problem under its real-life statement as there are multiple container types that should be considered, (refrigerated, open side, empty, dry, open top and tank). Often, containers arrive at the port dynamically over time and have an uncertain departure date (ship delayed, a ship down, delayed arrival of customer trucks…). Indeed, CSP must be studied in dynamic aspect.

The objective of this thesis is to study Static CSP for a single and various container type and dynamic CSP for ONE and several container types and to propose solutions for each of them. Genetic algorithm and Harmony Search algorithm are used to solve these problems and we compare the results of each approach with the LIFO algorithm

Keywords

Container stacking problem, Static, Dynamic, rehandling operations, metaheuristic, Genetic algorithm, Harmony search algorithm, optimization.